CMOS GATE DELAY, POWER MEASUREMENTS AND CHARACTERIZATION WITH LOGICAL EFFORT AND LOGICAL POWER

A Thesis
Presented to
The Academic Faculty

By

Richard B. Wunderlich

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Electrical and Computer Engineering

School of Electrical and Computer Engineering
Georgia Institute of Technology
December 2009

CMOS GATE DELAY, POWER MEASUREMENTS AND CHARACTERIZATION WITH LOGICAL EFFORT AND LOGICAL POWER

Approved by:

Dr. Paul Hasler, Advisor
Associate Professor, School of ECE
Georgia Institute of Technology

Dr. David V. Anderson
Associate Professor, School of ECE
Georgia Institute of Technology
Atlanta, GA

Dr. Saibal Mukhopadhyay
Assistant Professor, School of ECE
Georgia Institute of Technology
Atlanta, GA

Date Approved: November 11, 2009

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Paul Hasler, for giving me the flexibility to persue this topic, and for his patience while doing so.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The primary metrics associated with a logic gate's performance are speed, power, and area. We define a gate as a specific CMOS transistor level implementation of a particular boolean function in a specific fabrication technology at a constant rail voltage, constant length, and where the ratio of any two transistor widths are constant. Asking how fast a gate switches then is highly situational; it changes with load capacitance, choice of inputs, input slew rate, and the size of the gate. Predicting how much energy the gate consumes depends on the time frame, how many times the gate has switched in this time frame, input selection, input slew rate, load capacitance, and gate width. Logical Effort (LE) predicts gate delay with a simple linear equation: d = t(gh+p). Where g and p are gate and input dependent parameters independent of load size and gate size, and h is the ratio of output capacitance to input capacitance (directly related to gate width), and t is a process dependent conversion factor. The product, gh, then is the delay associated with driving a subsequent gate, and p is the delay of the gate driving itself. The prediction ignores input slew rate and the linear dependence fails for very large values of h, but for input slew rates on the same order as the output slew rate, and for reasonable fan-outs, LE provides remarkably accurate predictions of gate switching time. The methodology goes on to solve for the widths necessary for each gate in an arbitrary logic path to minimize delay. Designs can quickly be compared, analyzed and optimized. By breaking down delay into components, one is able to intuitively choose better logic implementations, if parasitic delay is dominating, often a better implementation is one with smaller fan-in gates and less logic depth, if effort delay is dominating then then higher logic depth can lead to faster results. What the method does not do is predict the power consumption ramifications of all of these choices. What about minimizing power on non-critical paths, for instance?

To our knowledge, no methodology exists to predict power consumption in a similar fashion. We propose a power prediction methodology, Logical Power (LP), compatible

with LE that breaks down power consumption into dynamic, static, and short-circuit components with linear equations dependent on h. This would allow a compact and efficient way to characterize a gate that scales with its environment, as well as to allow designers optimizing with LE to consider not only the speed ramifications of individual gate sizings but power as well. For instance given a target path delay higher than the theoretical minimum predicted by LE, sizings could be chosen with LE and LP that minimize power that still result in meeting the target delay.

The other major contribution of this work is a new short-circuit power measurement technique for simulation that more accurately distinguishes between short-circuit and the parasitic portions of dynamic power in total active power dissipation than all known techniques.

CHAPTER 1

INTRODUCTION

This work deals predominantly with the characterization of digital CMOS logic gates in order to predict delay and energy consumption. All models for delay and energy presented are naive, first-order, linear predictors. The model chosen to characterize and predict delay is the well known method of Logical Effort (LE). And this work proposes a method to predict power that is compatible with Logical Effort which we call Logical Power (LP).

Logical Effort breaks delay into two components, the *parasitic delay*, the delay associated with a gate driving itself, and *effort delay*, the delay associated with driving its load. The *effort delay* scales linearly with load capacitance and inversely with gate width. The rate at which delay increases with load capacitance is the *logical effort* of the gate. While the *parasitic delay* is independent of load size and gate sizing. Each gate has a unique *logical effort* and *parasitic delay*, independent of the gate's sizing and loading, that are used to predict its delay. The real power of the method comes in logic path delay optimization. Where given a specific logic path of gates, a set driving strength, and a set load capacitance, there exists a unique set of gate stage sizings that minimizes delay along the path. Logical Effort shows that delay is minimized not when the delay of each stage is equal, but when the *effort delay* of each stage is equated, that is, the delay of each stage less the sizing independent *parasitic delay*. The method can be further used to tell whether or not the ideal number of stages are being used, and subsequently whether or not additional inverters can be used to minimize delay even further.

Logical Power is proposed as a method to coincide with Logical Effort and predict the power ramifications of these sizing choices in the delay optimization problem. Keeping the same optimization problem scope as LE, that is, a handful of gates in a logic path and the method's desire to intuitive and conducive to hand analysis, LP is derived in a fashion that retains only the first-order linear terms in how a gate's power scales with output load size.

LP then predicts the *static power*, and *active power*, of a gate as a function of gate size and load size given a few gate dependent parameters.

A detailed analysis of the validity of the approximations of both methods is carried out, identifying where the methods are accurate and where they break down. How the methods scale with process size, and other design choices such as rail voltages and threshold voltages. As both methods ignore slew rate in their predictors, this will be one of the major assumptions analyzed.

A method for extracting the gate dependent parameters of LE and LP necessary to characterize a gate under these approximation techniques will be proposed. The effect of the input slew rate choices to the gate to be characterized will be shown as well as proposing a test circuit that gives the best average case results.

Lastly, both methods are verified for accuracy by predicting the power and delay of thousands of randomly generated logic paths comprised of randomly chosen gates and gate sizings with comparison to detailed spice simulations.

CHAPTER 2

DELAY

When dealing with CMOS logic gates where inputs are tied to the gates of transistors, the complex impedance looking into any one of the inputs is predominantly capacitive. As opposed to, for instance passfet logic, where an input can be tied to the source, drain, or gate of a transistor. The impedance looking into a source or drain can not be approximated accurately as simply capacitive. This work is restricted to logic gates where inputs are tied to the gates of transistors only, as is the case for CMOS logic.

When this is the case, the transient output of a gate has minimal effect on the load it provides to a gate driving it. Therefore one can express the total delay of gates in a chain as being the delay of the summation of the delays of the individual gates, Figure 1 . Where the delay of each individual gate is a strong function of load it sees on its output.

2.1 Measuring Delay

The delay predicted by *logical effort* is the 50% rail-to-rail input to output propagation delay of the gate. That is, delay is measured as:

$$d_f = t(v_o < V_{dd}/2) - t(v_i > V_{dd}/2) \tag{1}$$

$$d_r = t(v_o > V_{dd}/2) - t(v_i < V_{dd}/2) \tag{2}$$

$$d = d_{avg} = \frac{d_r + d_f}{2} \tag{3}$$

Where d_f and d_r are the falling and rising output propagation delays respectively, and the terminology $t(v_o < V_{dd}/2)$ means the time when the output first passes from logic level high to 50% rail-to-rail, etc. Figure 2 shows input and output voltage transient waveform shapes typical of fast operating CMOS gates driving CMOS gates. In the figure, the rising

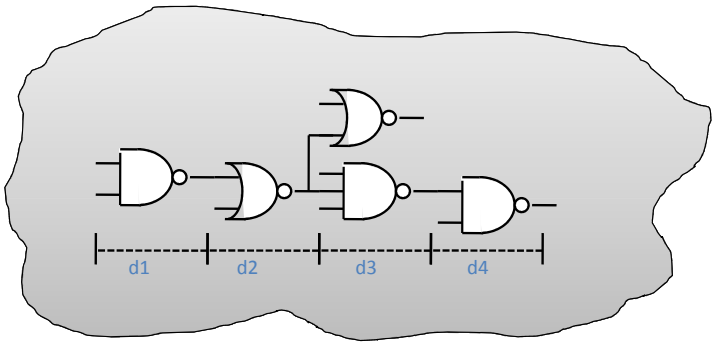

Figure 1. The delay of an arbitrary CMOS logic path as the summation of the individual delays of the gates.

propagation delay, d_r, is shown as well as the 90% to 10% output fall time, t_f, of the gate. The fall time, and the similar 10% to 90% rise time, t_r, are other important delay metrics associated with a gate, but this work will predominantly be concerned with the propagation delays.

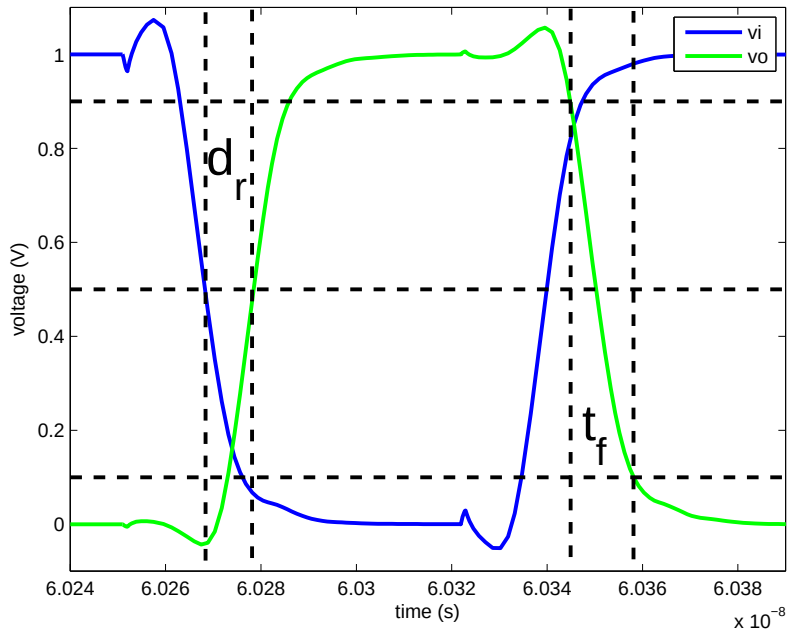

Figure 2. Input and output voltage transients of a CMOS gate driving a CMOS gate, shown are the measurements of rising propagation delay, d_r , and output fall time, t_f .

CHAPTER 3

POWER

3.1 Components of Power

Power dissipated in a logic gate is a complicated function of the gate's implementation and its environment. The logic family, transistor topology, transistor sizings, and rail voltages that all define a gate are only part of the picture. The load that the gate drives, and transient inputs all factor into the power a gate will dissipate over time.

The power dissipated by the gate can be broken down into three well defined components: *dynamic power, static power,* and *short-circuit power.* Dynamic power is simply defined as the amount of power consumed by charging and discharging capacitances seen by the logic gate and can be expressed as such:

$$P_{dyn} = \alpha f V_{dd}^2 C_{tot} \tag{4}$$

Where $\alpha$is the activity-factor defining the fraction of the cycles that the logic gate makes an output transition, f the cycle frequency, $V_{dd}$the rail-to-rail voltage, and $C_{tot}$the total amount of capacitance seen by the driving gate.

Whereas *dynamic power* is dissipated during an output transition, *static power* is the power consumed when the gate is not making a transition and is a simple function of the static current draw I_{stat}:

$$P_{stat} = V_{dd} I_{stat} \tag{5}$$

The last portion of power, *short-circuit* power*,* is defined as the portion of power consumed during an output transition that did not go to charging capacitances. This portion of power is quite difficult to express compactly, and for even the most trivial of logic gates, an inverter, has no closed-form analytical solution. Most approximations are extremely cumbersome. *Short-circuit power*, is a function of input slew rate, output slew rate, and gate

topology:

$$P_{s.c.} = \alpha f V_{dd} F(input - transition, output - transition) \tag{6}$$

3.2 Measuring Active Power

In order to extract the parameters of *Logical Power* we need a method to accurately measure the components of power for a given gate. To do so we observe various currents associated with a gate, illustrated in figure 3:

$$I_{GND} = I_b + I_{pd} + I_{bd} \tag{7}$$

$$I_{VDD} = I_w + I_{pu} + I_{bu} \tag{8}$$

Where I_{GND} is the total amount of current supplied by the GND rail, whose components are I_b, I_{pd}, I_{bd} being the currents leaving the bulk, pull down network, and any logic level zero bias applied to the circuit respectively. I_{VDD}being the V_{dd} rail equivalent.

The integrals of current are used to evaluate energy in the following relation:

$$E = Vq = V \int idt \tag{9}$$

Where the integral is taken over a time period that returns the gate to its starting state. This ensures that all flux that enters through the positive terminal of the voltage leaves through the negative terminal with no net flux accumulating in the circuit. That is, that all charge entering the circuit through V_{dd} leaves through GND. All power measurements are done through explicit integration of saved values of currents post simulation, as opposed to using power-meter sub circuits like the one proposed in [1] and used in [2].

Integrals of $I_{VDD}$and I_{GND} for a rising then falling transition for an arbitrary inverter in Figure 4. In this plot there are four distinct regions of interest. In Region I there is a significant amount of charge leaving through V_{DD} corresponding to a rising transition

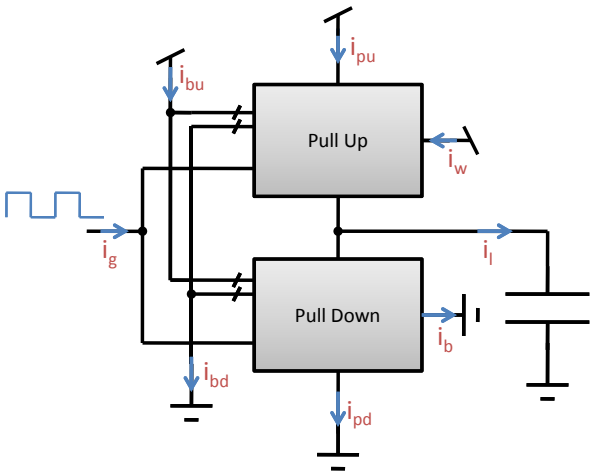

Figure 3. An arbitrary CMOS logic gate with pull up and pull down networks and corresponding currents

at the output of the gate. This current goes to charging capacitances and thus does not immediately show up through *GND* network of the gate. Region II is after the rising transition has settled and the gate has entered a static region, with Region III being the falling transition and Region IV the post falling static mode. One can see that after the gate has been restored to its original state does all charge that has entered the circuit leave the circuit.

Active energy is then extracted by integrating this current (either i_{vdd}or i_{gnd}, both being equivalent) over Region I and Region III:

$$E_{act} = V_{dd}\left[\int_{t1}^{t2} i_{vdd}dt + \int_{t3}^{t4} i_{vdd}dt\right] \tag{10}$$

In order to break active power down into its components, *short-circuit power*, and *dynamic power*, some choices need to be made on how to handle an issue known as voltage overshoot.

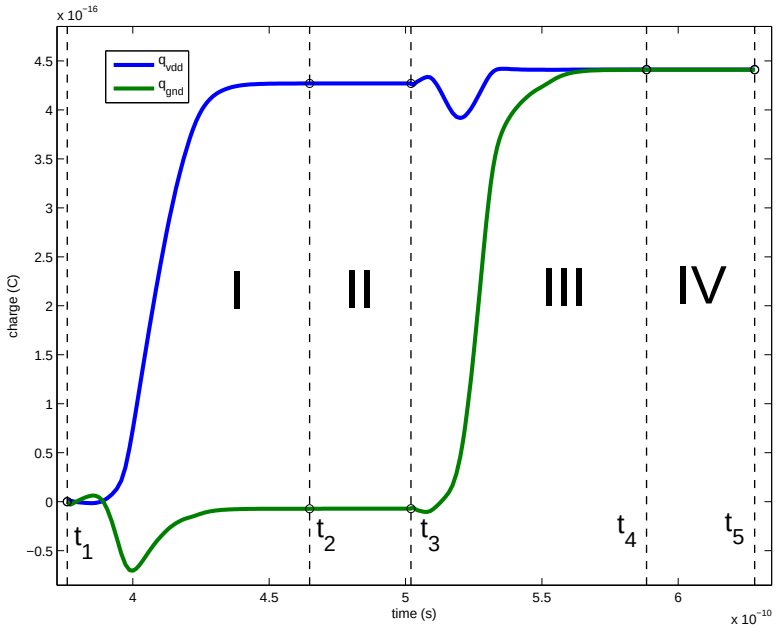

Figure 4. Integrals of current through the V_{dd} and GND rails

3.3 Voltage Overshoot and Miller Effect

The transient output voltage of a CMOS gate can go above the V_{DD} and below the GND for that particular gate during transitions as can be seen in Figure 5. This phenomena, called voltage overshoot, is due to capacitive coupling of a gate's input to its output by the gate to drain capacitances of the transistors.

Consider a falling output transition for a logic gate with the input voltage initially at GND. The output voltage is initially settled to V_{DD} when the input voltage starts to rise. If the capacitive coupling between the input and output is called C_M and the total capacitance seen at the output is C_L, then for instantaneous changes in the input voltage equal to $\triangle V_{in}$ the output voltage will go to:

$$V_{out} = \triangle V_{in} \frac{C_M}{C_L} + V_{DD} \tag{11}$$

The magnitude of the voltage overshoot depends on the input slew rate and the speed by which the output can discharge. During the voltage overshoot the normal polarity for

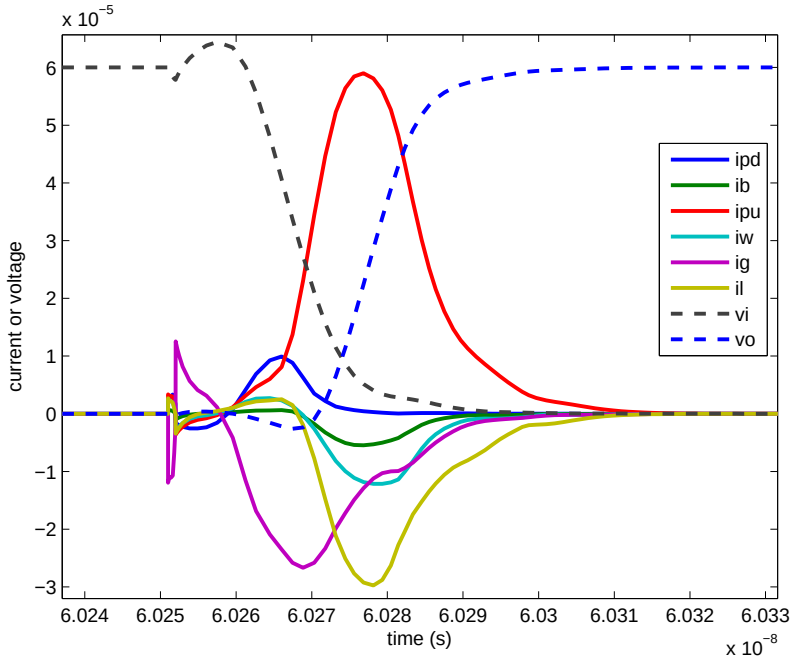

Figure 5. Transient voltages and currents during a rising output transition showing voltage undershoot at the output as well as the subsequent negative current flowing into GND.

V_{DS} of the pull up network reverses and the output is helped to discharge through the pull up network with a positive current flowing into V_{DD}. Power is dissipated in the pull up network due to this that is not short-circuit power, in fact, this phenomena seems to reduce short-circuit power. Total power consumption is not increased by this effect, it just moves a fraction of the driving gate's dynamic power to be dissipated in the load gate.

To see this, consider the analogous circuit in Figure 6. The values of R_{U1} , R_{U2}, C_m's, V_{DD1} and V_{DD2} from both circuits are equivalent to each other. In the first circuit R_{U1} models the pull up or pull down network resistance of the driving gate, with the switching between the two networks being modeled by the voltage source V_{DD1} pulsing between V_{DD} and $0V$. The capacitance C_m models the input capacitance to the load gate with $C_m + C_L$ being the total capacitance seen on the output of the load gate. The output voltage of the load gate is V_{o1}, this is where the voltage overshoot will occur, and R_1 models an arbitrary discharging path for this overshoot (after a rising or falling input, the output is always discharged to

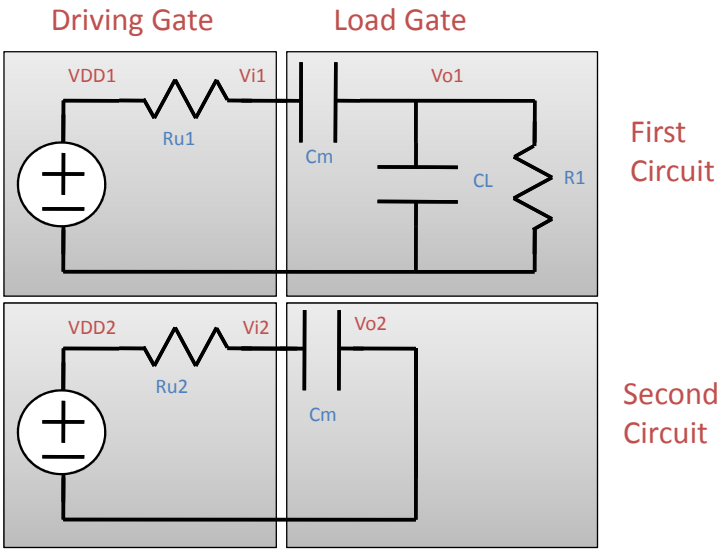

Figure 6. Example circuits for demonstrating the principles of voltage overshoot in CMOS gates. In the second circuit, V_{i2} corresponds to the input voltage to a load gate with no capacitive coupling to the load gate's output, whereas in the first circuit the input voltage V_{i1} is coupled to the load gate's output voltage V_{o1}.

ground, which is not the case for CMOS gates, none-the-less this circuit is sufficient to show the trends on how output voltage swing can affect power and delay). The second circuit models this effect for when overshoot is reduced to insignificance by either C_L or R_1 being sufficiently large or small with respect to the other circuit elements. The transient voltage behavior is shown in Figure 7 showing both the voltage overshoot above V_{DD} during the rising input transition and below GND during the falling input.

Inspection of the second circuit under conservation of energy suggests that when V_{DD2} increases to V_{DD}, the voltage V_{i2} eventually charges to V_{DD} storing an amount of energy on C_m equal to $0.5C_mV_{DD}^2$ with an equivalent amount of energy having been dissipated in R_{U2}. When V_{DD2} is brought back down to $0V$, V_{i2} eventually reduces to $0V$ with all of the potential energy stored in C_m discharged through R_{U2} . Total energy dissipation of both transitions being $C_mV_{DD}^2$ and all of it dissipated in R_{U2}.

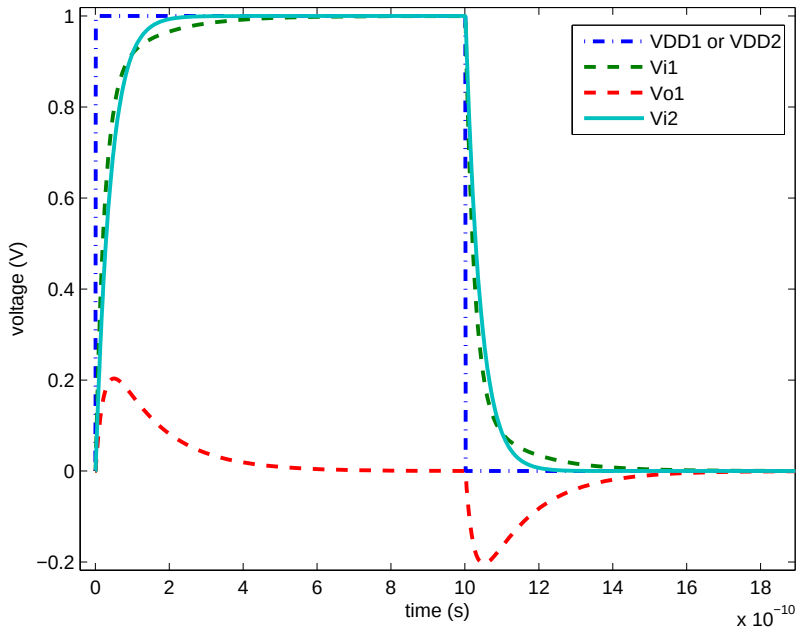

Figure 7. Transient voltages for both circuits. The in

While the first circuit is much harder to solve out exactly, one can see that the final amount of energy stored in the circuit upon settling after V_{DD1} goes to V_{DD} is the same as in the second circuit, $0.5C_mV_{DD}^2$. And it turns out that exactly this amount of energy is dissipated in the circuit's resistors during charging and dissipated again during discharging. That is, total energy consumption of both transitions is the same as the second circuit, but now the energy is dissipated across two resistors, one being in the analogy of the load gate. The total energy dissipation of both circuits as well as the dissipation in each resistor is shown in Figure 8.

The analogy then suggests that voltage overshoot does not increase dynamic power dissipation above what is already incurred due to load capacitance, as in both cases the energy dissipation was defined by C_m only. The phenomena only moves a fraction of the energy dissipation into the resistances of the load gate. The magnitude of the fraction being related to the magnitude of the voltage overshoot.

Even though this produces no affect on the total dynamic power, it can have a significant

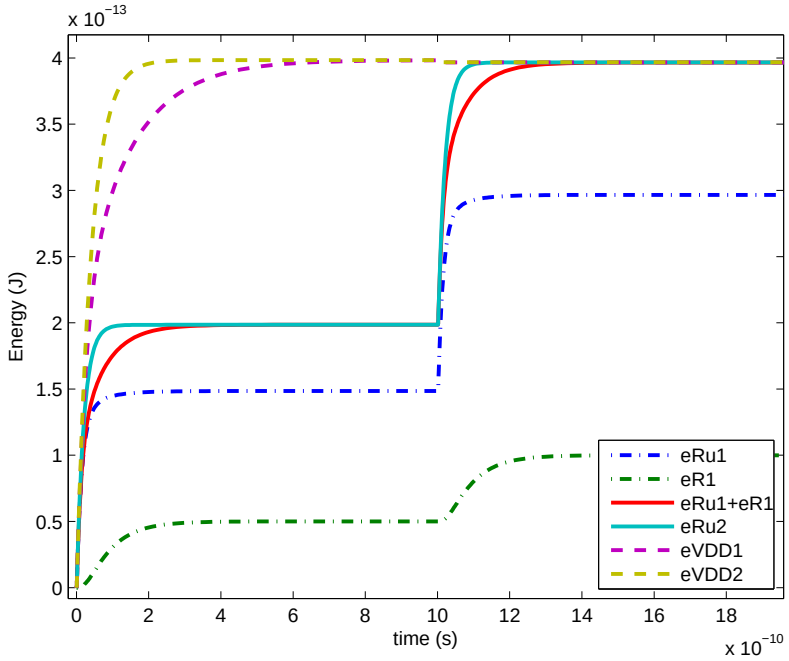

Figure 8. Energy dissipated by resistances and sourced by voltage sources as a function of time for the two different circuits for a rise and a fall. As time tends towards infinity, the energy dissipated in both circuits is identical.

affect on delay. To observe this, notice that the delay to 50% rail-to-rail of V_{i1} is less than that of V_{i2} as can be seen in Figure 9 . This is because in the first circuit, over this range of observation, the output voltage tracks the input voltage, effectively reducing the voltage across the input capacitance C_m and reducing the amount of charge necessary for a given change in input voltage. Whereas, in the second circuit the output voltage is fixed. The net effect being that the delay over this range is less for the first circuit.

This effect on delay can be explained by the Miller Effect on the effective input capacitance, $C_{effective}$, looking into the load:

$$C_{effective} = C_m(1 - A_v) \tag{12}$$

Where C_m is the capacitance coupling the input to the output, and A_v is the gain from the input to the output. In the example of circuit one, the gain is positive and less than one over this range, and thus the effective capacitance is less than the case of the second circuit

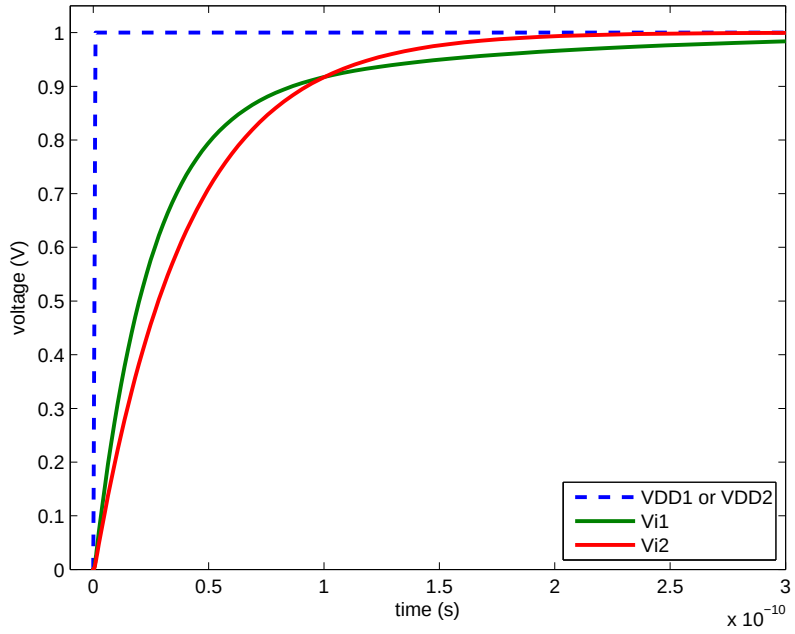

Figure 9. The Miller Effect on delay.

with zero gain. For times larger than this range, where the output voltage stops tracking the input voltage, the effective gain becomes negative, and an increase in effective capacitance is observed. Therefore, the affect of the Miller Effect on the delay of the driving gate depends highly on the output slew rate of the load gate, and therefore on the load seen by the load gate.

For standard CMOS circuits, depending on the load's load size, the Miller Effect can either increase or decrease propagation delay. For reasonable load sizes, like shown in Figure 5, the net effect is trivial. However, if the load gate was not loaded, the output voltage would swing much faster, and could cause a significant overlap of V_{out} and V_{in} which would increase the gain over this region and thus the effective input capacitance, which in turn would increase delay.

This means that in the characterization of a logic gate for power, one can safely ignore the affects of the Miller Effect and voltage overshoot, whereas this is not quite the case for delay characterization.

3.4 Measuring Short-Circuit Power

Short-Circuit power contributes an additional energy cost to active gate transitions. It arises from increased conductivity between V_{DD} and GND during an output transition due to both pull up and pull down networks being in between their high and low impedance states. Accurately measuring short-circuit power dissipation in simulation is complicated by capacitive currents flowing in the turning-off network during transition.

Most methods for measuring short-circuit power, the methods used in [3, 4, 2] for instance, involve measuring the current the charge flowing through the turning-off network during a transition and assuming that all of this charge contributes to short-circuit power dissipation. The problem with this assumption is that that simply is not the case. One of the contributions of this thesis is to show that a rather significant portion of this current is due to capacitive charging and discharging in the turning-off network that should be attributed to dynamic power and not short-circuit power, as well as to propose and analyze a method of measuring short-circuit power that fixes this problem.

$$E_{sc} = V_{dd}\left[\int_{t1}^{t2} i_{gnd}dt + \int_{t3}^{t4} i_{vdd}dt\right] \tag{13}$$

Figure 10 shows the major components of a MOSFET as modeled by the EKV v2.6 device model [5]. For this section we are only concerned with I_{DS}, the DC channel current, and the total effective terminal-to-terminal capacitances C_{GS}, C_{GD}, C_{SB}, C_{DB} and ignoring the gate to bulk capacitance and various diodes and other current sources. Figure 11 shows an inverter with these MOSFET capacitances explicitly drawn.

Consider a rising output transition. Previous to the transition the pull down network provided a low impedance path to GND, the pull up network a high impedance path to V_{DD}. After the transition these paths are swapped and a strong path to V_{DD} is connected to the output while the path to GND is weakened. During the transition, however, both networks enter intermediate states that can allow for relatively strong connections from V_{DD} and GND to be connected to the output, providing a low impedance path between

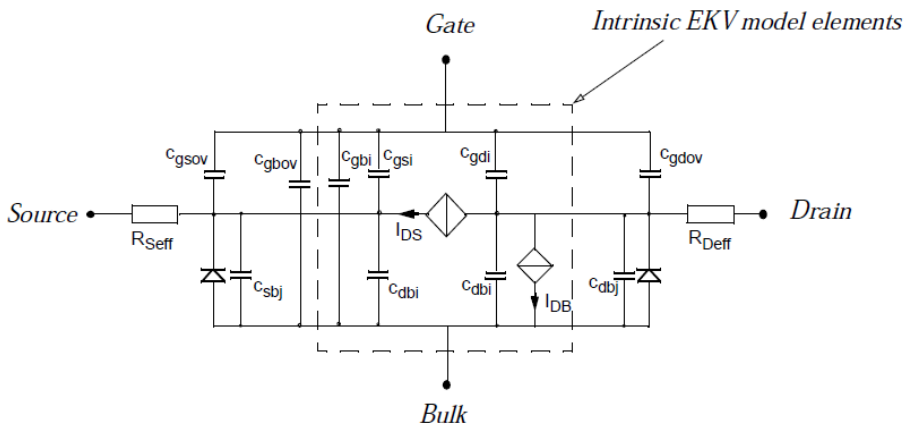

Figure 10. The intrinsic and extrinsic elements of a MOSFET modeled in the EKV v2.6 device model. It is the portion of the Source and Drain terminal currents not coming from capacitances that should be used for calculating short-circuit power.

both voltage rails. Which causes current to flow through both networks creating short-circuit power dissipation.

Let i_{pd} be the current going into GND from the pull down network minus the bulk current, i_b. Define q_{pd} and q_b to be the integrals of current with respect to time over the rising transition, and q_{pu} and q_w to be integrated over the falling transition. If $q_{sc-rise}$ and $q_{sc-fall}$ are the total amounts of charge flowing through the gate due to short circuit during a rise and fall transition respectively, then the total energy dissipation of a rise and fall transition due to short-circuit power is:

$$E_{s.c.} = V_{DD}(q_{sc-rise} + q_{sc-fall}) \tag{14}$$

Considering the rising output transition still, the problem is to relate $q_{sc-rise}$ to q_{pd} and q_b. However, both of these quantities contain charge that flowed onto and off of transistor capacitances. For instance, a large portion of q_{pd} is due to the negative current flowing out of GND from the discharging of the gate to source capacitance of the nFET by the input transitioning low. Capacitive feed through from the input to the output causes the output to go below GND for a small portion of time creating a negative V_{DS} on the nFET and causing current to flow from GND to the output node. Neither of these types of contributions to

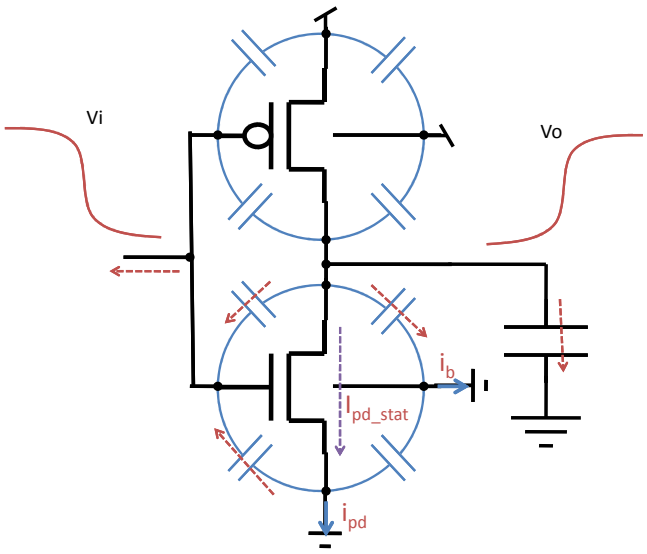

Figure 11. Differentiating between capacitive and static currents during a rising output transition.

the current should be considered as short-circuit power as they do not fit the definition of conduction from rail to rail.

In [6] they attempt to pull these currents out of the problem by integrating the total positive current flowing into *GND* for a step input and subtracting this out of subsequent measurements for non step inputs:

$$f(i) = \begin{cases} i & i > 0 \\ 0 & i \leq 0 \end{cases} \tag{15}$$

$$E_0 = V_{dd}\left[\int_{t1}^{t2} f(i_{gnd})dt + \int_{t3}^{t4} f(i_{vdd})dt\right] \ldots v_{in}(t) = V_{dd}[u(t-t_1) - u(t-t_3)] \tag{16}$$

$$E_{sc} = V_{dd}\left[\int_{t1}^{t2} f(i_{gnd})dt + \int_{t3}^{t4} f(i_{vdd})dt\right] - E_0 \tag{17}$$

Where E_0 is the short-circuit measurement for step inputs that gets pulled out of measurements of E_{SC} for non step inputs.

This work proposes a more direct measurement of short-circuit power. Considering the MOSFET model shown in Figure 10 there is a total amount of current flowing into the source or drain that is due to MOSFET capacitances and a portion due to the DC channel current i_{DS}. By integrating the positive portion of this i_{DS} of the switching transistor in the turning off network we directly measure total short-circuit charge without having to manually deal with dynamic and feed through charge effects. Measurements done in this manner inherently converge on zero (neglecting sub threshold leakage currents, etc.) for increasingly fast input transitions.

$$E_{sc} = V_{dd}\left[\int_{t1}^{t2} f(i_{pd-stat})dt + \int_{t3}^{t4} f(i_{pu-stat})dt\right] \tag{18}$$

Where $i_{pd-stat}$ is the static current coming out of the source of the switching transistor in the pull down network, $i_{pu-stat}$ the static current entering the source of the switching transistor in the pull up network, and $f()$ is the same rectifying function as before.

Being able to differentiate between capacitive and static currents depends on whether or not the simulator provides access and whether or not the model keeps track. For instance, in the SPECTRE circuit simulator, the static and capacitive currents going into the source of a transistor, $M0$, can be accessed as:

```
save M0:s:static M0:s:displacement
```

A comparison of the short-circuit measurement methods is shown in Figure 17. An inverter was simulated with increasing input slew rates for varying amounts of fanout. The black line in the figure shows input slew rate divided by average propagation delay (generated by that input slew rate) for a FO4 inverter. In each simulation, measurements of short-circuit energy were computed using the three different methods described in this section. Method 1 being the integral of GND current during a pull up transition plus the integral of V_{DD}

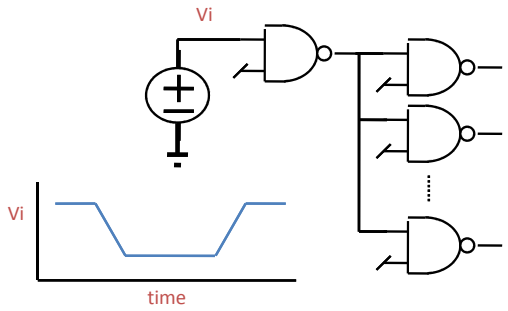

Figure 12. Test circuit with variable input slew rate and *electrical effort*.

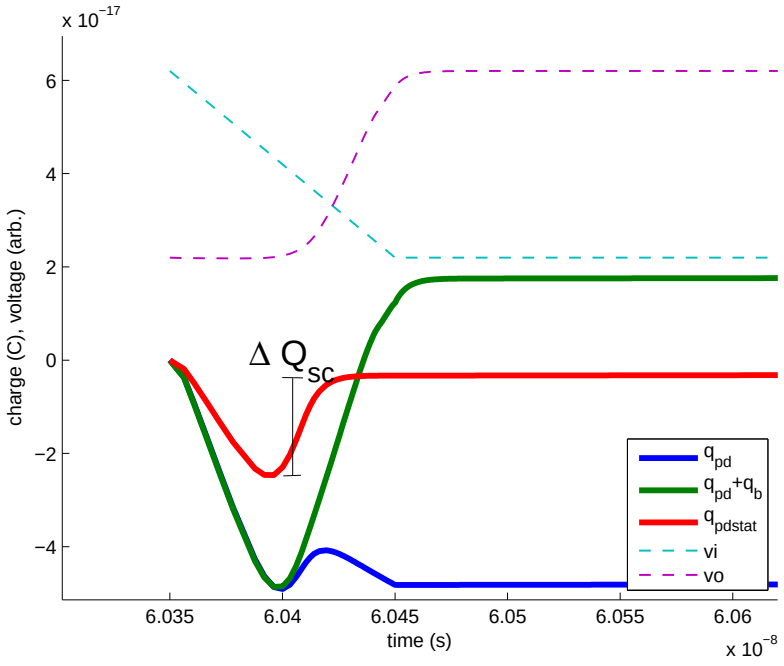

Figure 13. Integrals of current in the pull down network during a rising output transition.

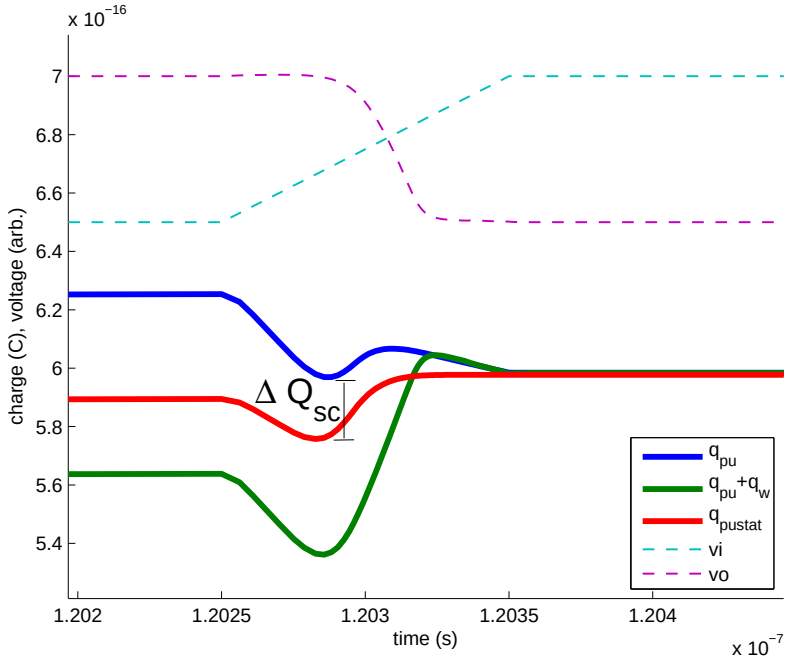

Figure 14. Integrals of current in the pull up network during a falling output transition.

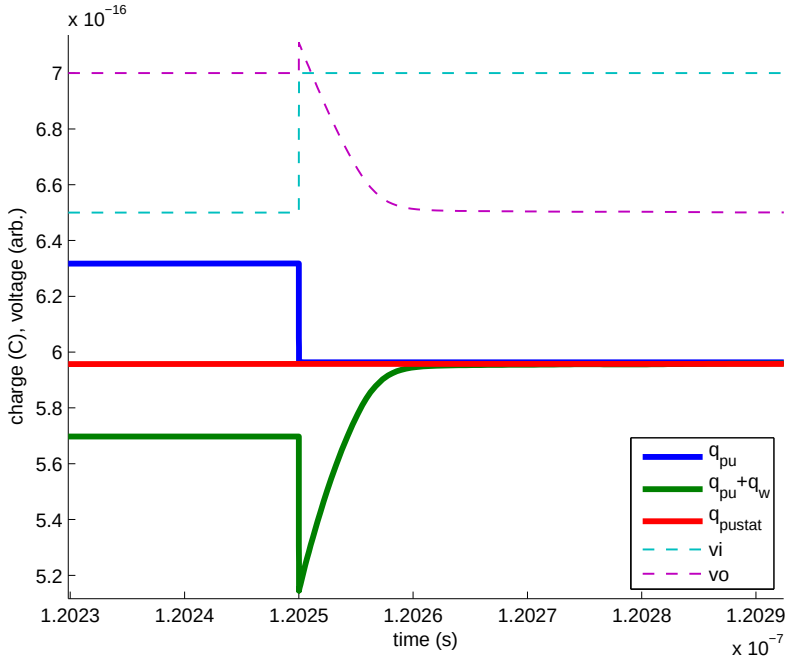

Figure 15. Integrals of current in the pull up network during a very fast rising input.

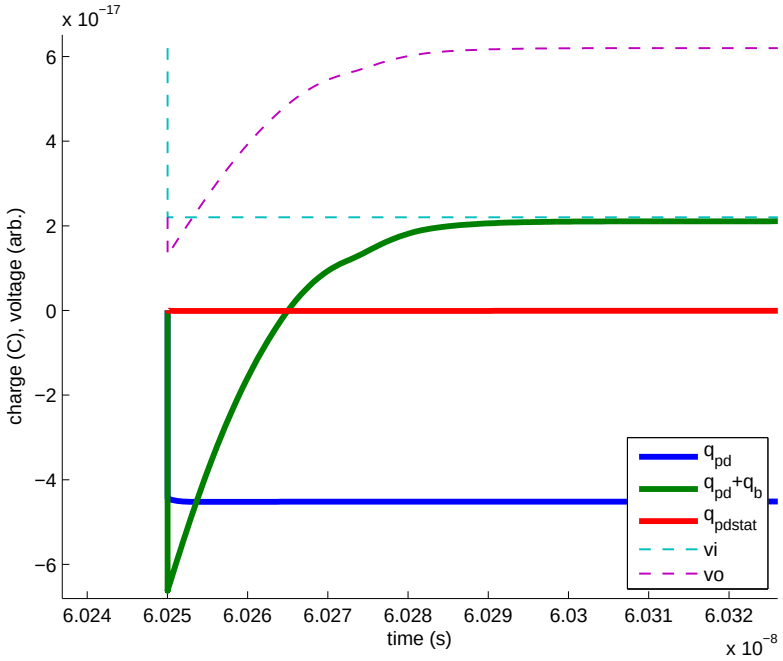

Figure 16. Integrals of current in the pull down network during a very fast falling input.

current during a pull down as described by Equation 13 . Method 2 being the integrals of only the positive portions of these currents minus the value obtained for a step input, Equation 17. And Method 3, the one proposed in this thesis, where only the positive static currents in the off-switching transistors are integrated as described by Equation 18.

Method 1 does not trend to zero for increasingly fast inputs. Method 2 does but only by manually setting this energy to be zero for step inputs. Method 3 obtains zero naturally. Methods 1 and 2 both have negative trends initially for short circuit energy with increasing slew rates. This produces negative answers with Method 2 for a large range of slew rates. Intuitively, short-circuit energy should monotonically increases for slower input slew rates, converging on zero for increasingly fast slew rates, and monotonically decreases with output load size (higher load capacitance leads to slower output transitions and thus less overlap between input voltage and output voltage). Only Method 3 satisfies all of these trend expectations. Figure 18 shows the various methods applied in the exact same fashion (only the FO4 case is shown) to a nand2 gate's b-input (the input connected to the gate

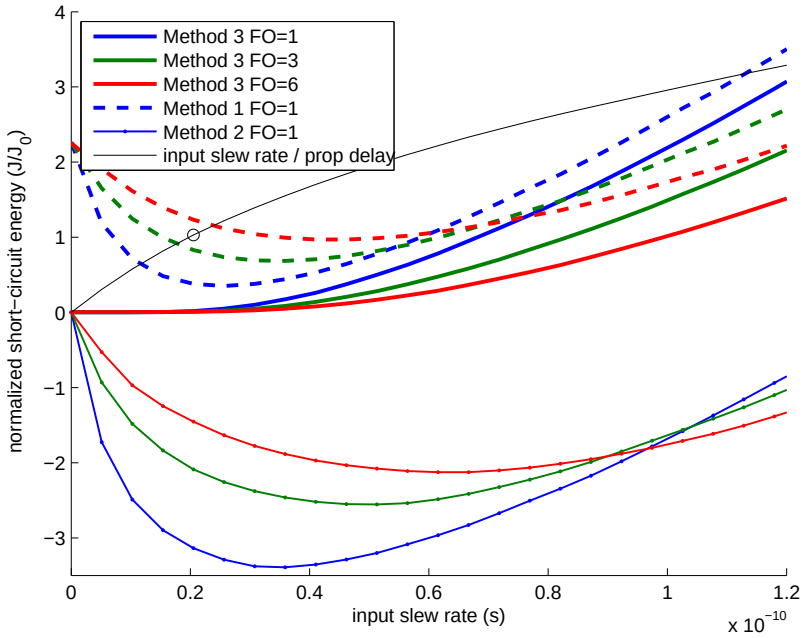

Figure 17. Comparison of short-circuit energy measurement techniques for an inverter for varying input slew rates and various fanouts.

of the nFET farthest from the output). Here Method 1 gets farther off as this gate has more parasitic capacitance to charge than the inverter case, and Method 1 is incapable of differentiated the parasitic capacitance charging from short-circuit energy.

3.5 Measuring Dynamic Power

Dynamic energy is simply the difference of *active* and *short-circuit power*:

$$E_{dyn} = E_{act} - E_{sc} \tag{19}$$

Dynamic power is not measured directly, it is simply the short-circuit power measurement subtracted out of the active power measurement (something that is assumed to be trivial to measure), the correctness of the dynamic power extraction then depends on the quality of our measurement of short-circuit power. Fortunately, we can at least make sure

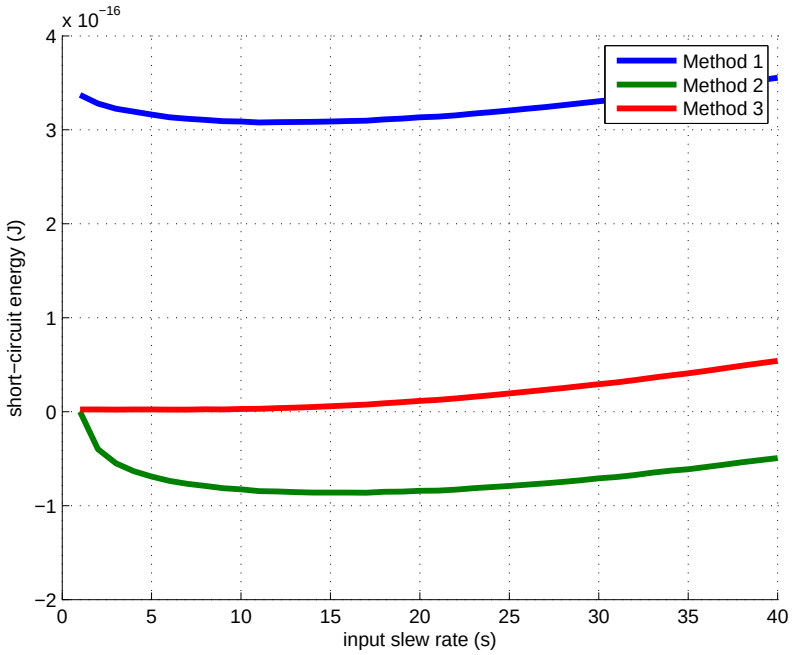

Figure 18. Comparison of short-circuit energy measurement techniques for the b-input (transistor farthest from output) of a FO4 nand2 gate for varying input slew rates.

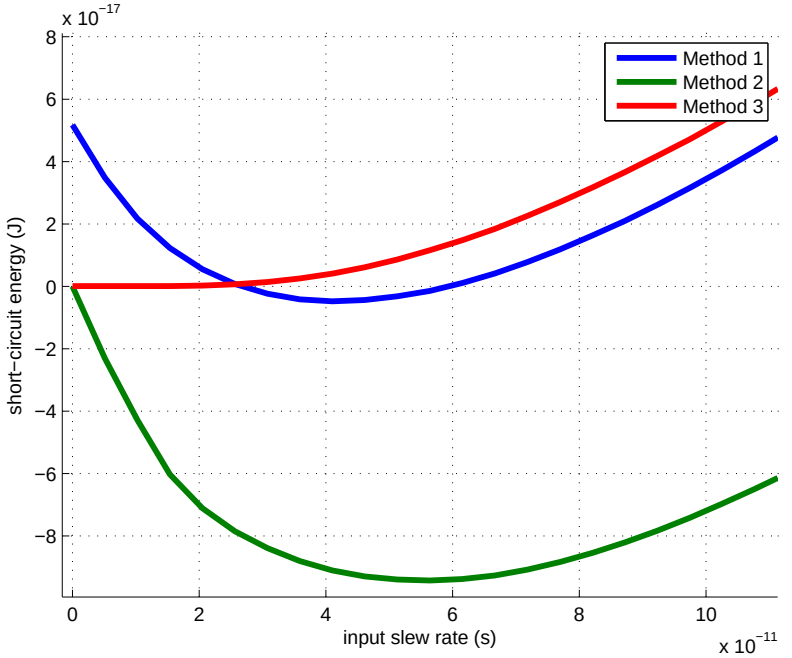

Figure 19. Comparison of short-circuit energy as measured by the different techniques for a FO1 inverter sized roughly for equal rise and fall times. Here it can be seen that Method 1 can give negative answers as well.

that our measurements of short-circuit power are causing dynamic-power to scale as expected.

We do that by observing our definition of E_{dyn} and its dependence on h, and taking the derivative of E_{dyn} with respect to h:

$$E_{dyn} = C_{tot}V_{dd}^2 = V_{dd}^2(C_{out} + C_p) = V_{dd}^2(C_{in}h + C_p) \tag{20}$$

$$E_{dyn,vh} = \frac{d}{dh}E_{dyn} = V_{dd}^2 C_{in} \tag{21}$$

Where $h = C_{out}/C_{in}$ is called the *electrical effort* of the gate and is related to the fanout. The quantity h can be easily controlled by having the gate driving multiple copies of itself, where the number of copies is equal to the *electrical effort* .

Then by taking *dynamic power* to be the difference of *active* and *short-circuit*. We arrive at a way to verify whether or not *dynamic power* is scaling linearly with output load capacitance, but not whether or not we are confusing *short-circuit power* with the portion of *dynamic power* associated with parasitic capacitance:

$$Err = 100\left|\frac{\frac{E_{dyn,vh}}{C_{in}} - V_{dd}^2}{V_{dd}^2}\right| \tag{22}$$

This error metric is also dependent on the measurement of the input capacitance to the gate:

$$C_{in} = \frac{-1}{V_{DD}}\int_{t1}^{t2} i_g dt = \frac{1}{V_{DD}}\int_{t3}^{t4} i_g dt \tag{23}$$

Where the limits of integration are defined by the *Regions* of Figure 4 as per usual, the first integral being over *RegionI* which is a rising output transition and therefore current is flowing off of the driving gate, and in the second integral taken over *Region*3 the output is undergoing a falling transition and therefore current is flowing onto the gate. This total

Table 1. Simulation parameters and tolerances

name	value
iabstol	1e-14
vabstol	1e-6
reltol	1e-4
gmin	1e-12
method	gear2only
simulator	SPECTRE MMSIM70
model	BSIM 4v4

charge flowing onto or off of the gate in either region should be equal and magnitude, it divided by V_{DD} is the effective capacitance.

Results for an inverter versus input slew rate can be seen in 20 for the various different *short-circuit* measurement methods as described in Section 3.4 . In general this will also depend on how well SPICE does at conserving charge and is then dependent on quite a few SPICE parameters. For all simulations reported in this document, the following parameters were the same, and listed in table 1.

It can be seen that Method 1, though incapable of distinguishing the parasitic portion of *dynamic power* from *short-circuit* gets the scaling of *dynamic* with load capacitance (in this case) better than the other two methods. Though, none of the methods have a particularly high error in this metric over the entire range. As a useful point of reference, Figure 21 shows normalized measurements of input gate capacitance, C_{in}, versus slew rate for various fanouts (in this case the fanout is equal to the *electrical effort*). It is expected that this quantity be constant, but instead changed with the same relative magnitude by which Methods 2 and 3 are erroneous in the *dynamic* scaling metric.

Figures 2223 show the same simulations but this time for the b-input (the input connected to the gate of the nFET farthest from the output) of a 2-input nand gate. For this more complicated gate, it can be seen that the measured C_{in} is much more consistent and that Method 3 produces the least error.

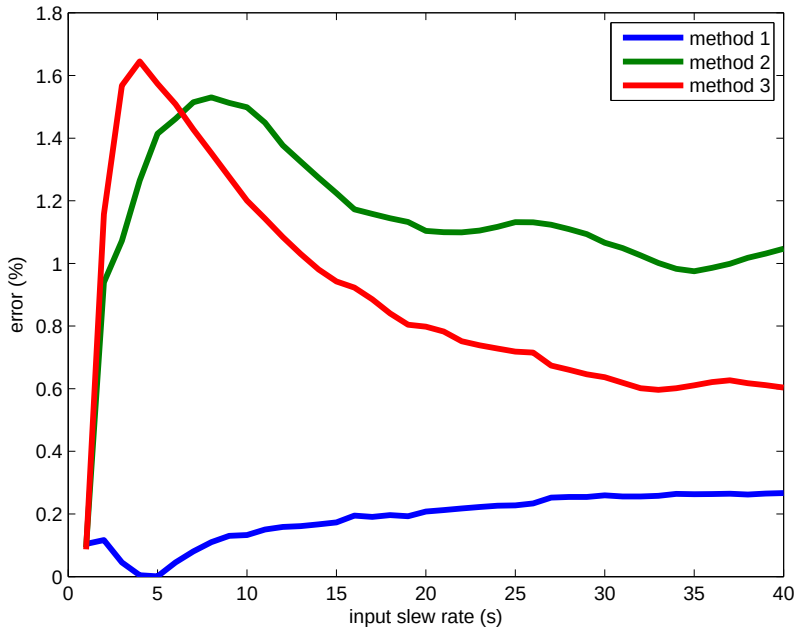

Figure 20. Error in expected *dynamic power* scaling (Equation 22) as extracted from *active power* by varying *short-circuit* measurement methods. All simulation points are of an inverter of $h = 1$ versus input slew rate.

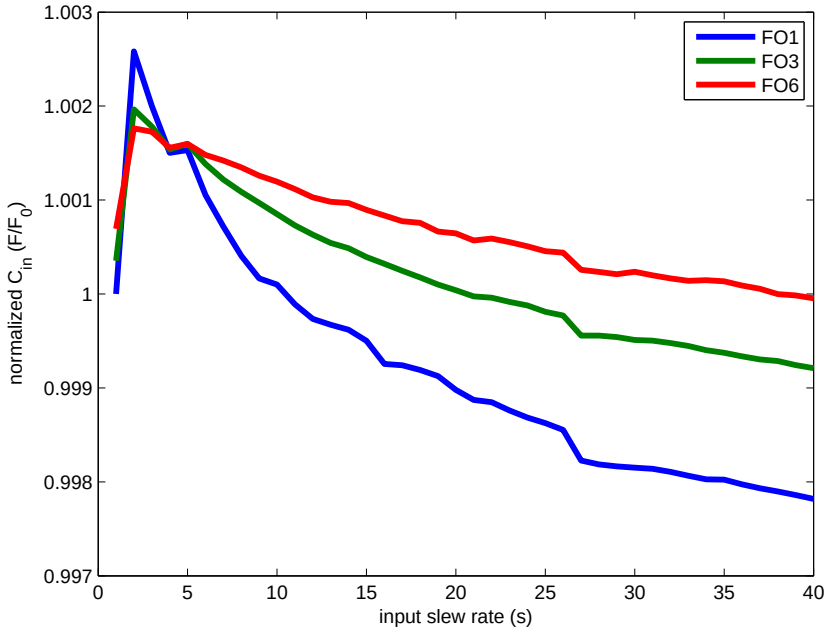

Figure 21. Measured effective input capacitance, C_{in} (Equation 23). All simulation points are of an inverter of $h = 1$ versus input slew rate.

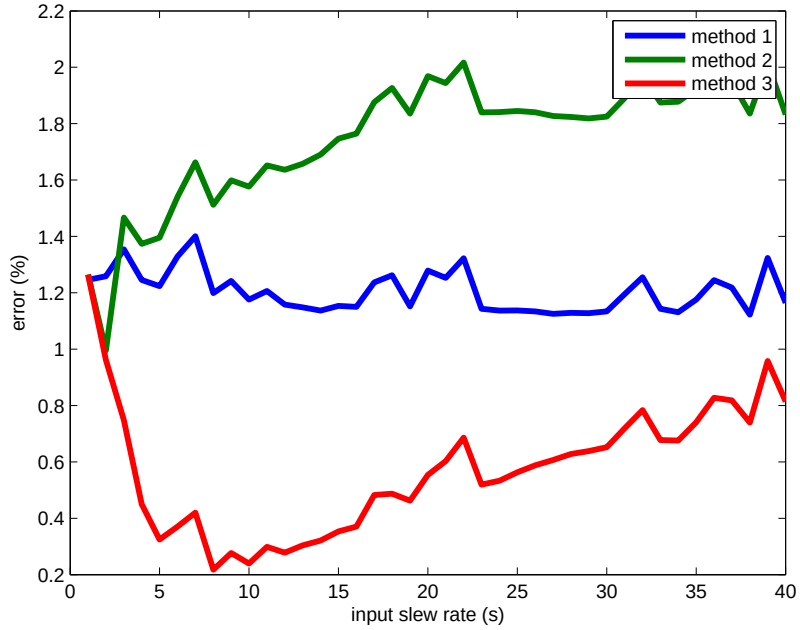

Figure 22. Error in expected *dynamic power* scaling (Equation 22) as extracted from *active power* by varying *short-circuit* measurement methods. All simulation points are of the b-input (input farthest from output) of a 2-input nand gate with $h = 1$ versus input slew rate.

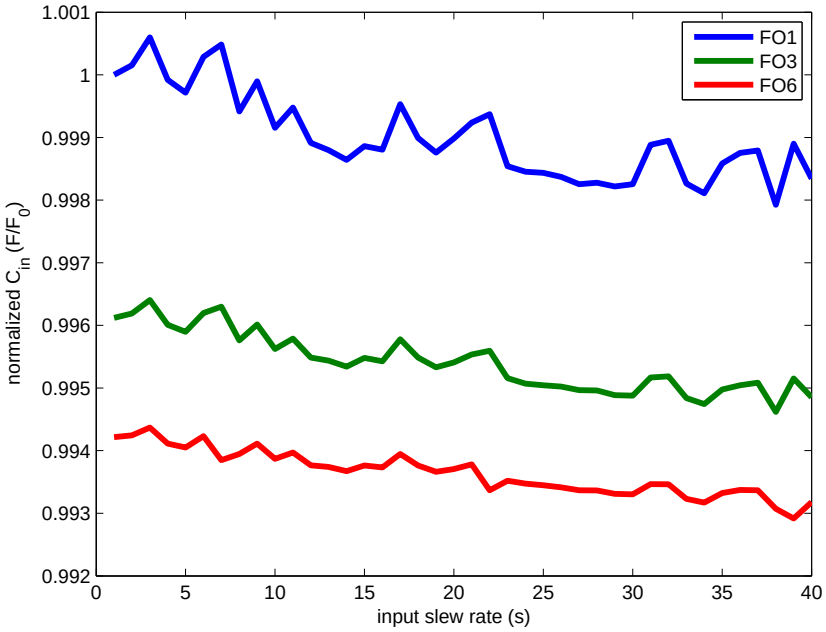

Figure 23. Measured effective input capacitance, C_{in} (Equation 23). All simulation points are of the b-input (input farthest from output) of a 2-input nand gate with $h = 1$ versus input slew rate.

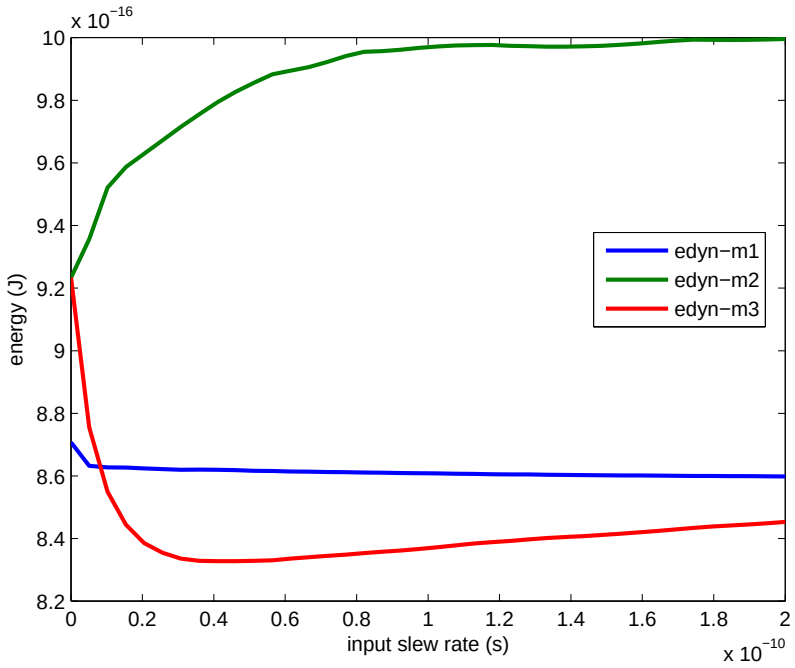

Figure 24. Dynamic energy as extracted from active energy from the various short-circuit methods for a FO1 inverter versus input slew rate.

3.6 Measuring Static Power

Assuming insignificant amounts of gate and load current in Regions II and IV, *static power* is simply:

$$P_{stat,avg} = \frac{V_{dd}}{2}[i_{vdd}(t \in RegionII) + i_{vdd}(t \in RegionIV)] \tag{24}$$

Where $i_{vdd}(t \in RegionII)$ means the current flowing out of the the V_{DD} rail during some time, t, in $RegionII$, etc. With $RegionII$ and $RegionIV$ being the static regions of operation after a pull up and a pull down transition as defined in Figure 4.

However, in modern processes, static gate current is a significant source of of leakage power and needs to be included in P_{stat} measurements for accuracy. This would at first appear to cause a significant dependence on load size for leakage current. Larger load sizes, when comprised of CMOS gates, mean a larger amount of gate leakage current in the load gates that has to be sourced (after a rise) and sinked (after a fall) by the driving gate.

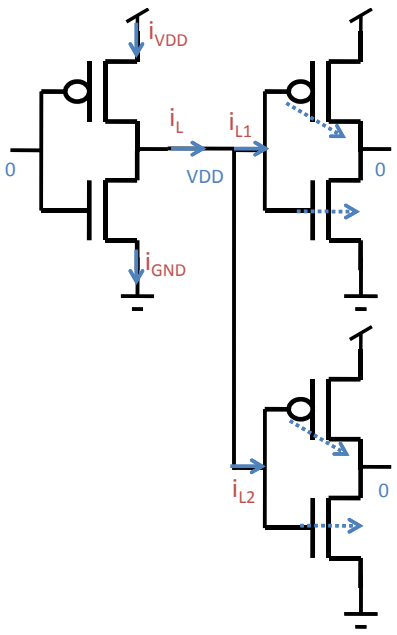

Figure 25. Gate leakage current paths for a FO2 inverter after a rise transition.

Consider the FO2 inverter of Figure 25 . The driving gate has finished making a rise transition, and its output has charged to V_{DD}. Static currents flow through all terminals of the gate: subthreshold currents in the off transistors in the pull down network, gate currents flowing out of the logic gate, and load current flowing into the gates of the load logic gates. This current flowing into the load depends on the load topology, and scales linearly with fanout.

In this case Equation 24 would overestimate the amount of static power consumed by this gate, as it simply measured the amount of current leaving the V_{DD} rail and assumes that this current passes to GND in this gate. Which is not the case, as the current flowing into the load is just a source for the gate leakage power dissipation of the load gates and does not get dissipated in the driving gate. To properly assign leakage power dissipation to gates in a topology like this, it becomes much easier to make approximations about the voltage drops across the pull up and pull down networks.

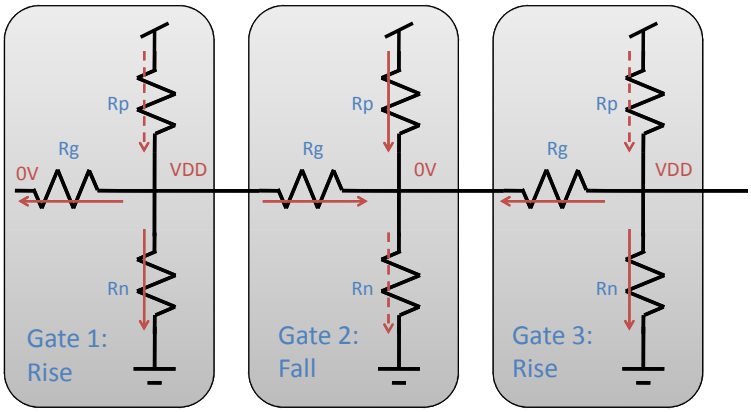

Figure 26. Static currents in a 3-stage path. In each gate the pull down network, pull up network, and gate network are approximated as R_p , R_n , and R_g respectively. Solid red arrows show currents dropping a voltage of VDD, whereas dashed red arrows represent currents flowing through no appreciable voltage drop. Static power then is measured as $P_{stat} = -V_{dd}[i_{gnd} + i_g]$ for a gate that has risen and $P_{stat} = V_{dd}[i_{vdd} + i_g]$for a gate that has fallen. With all currents referenced going into the logic gate.

Assuming that after a rise, that the output is charged near enough to V_{DD} that the voltage drop across the pull up network is negligible, then we can approximate that no leakage power is dissipated in the pull up network. And after a fall approximate that no leakage power is dissipated in the pull down network. Leakage power then becomes:

$$P_{stat,rise} = -V_{DD}[i_{gnd} + i_g] \tag{25}$$

$$P_{stat,fall} = V_{DD}[i_{vdd} + i_g] \tag{26}$$

Where $P_{stat,rise}$ is the static power dissipation after a rise ($RegionII$) and $P_{stat,fall}$ is the static power dissipation after a fall ($RegionIV$). Average static power dissipation of a gate driving multiple gates can then be expressed as:

$$P_{stat,avg} = V_{DD}[(i_{vdd} + i_g)|_{t\in RegionIV} - (i_{gnd} + i_g)|_{t\in RegionII}] \tag{27}$$

Figure 27 shows various ways to measure leakage currents in a gate with fanout. From this figure it can be seen that measuring leakage power by Equation 27 has the desired effect of being independent of fanout.

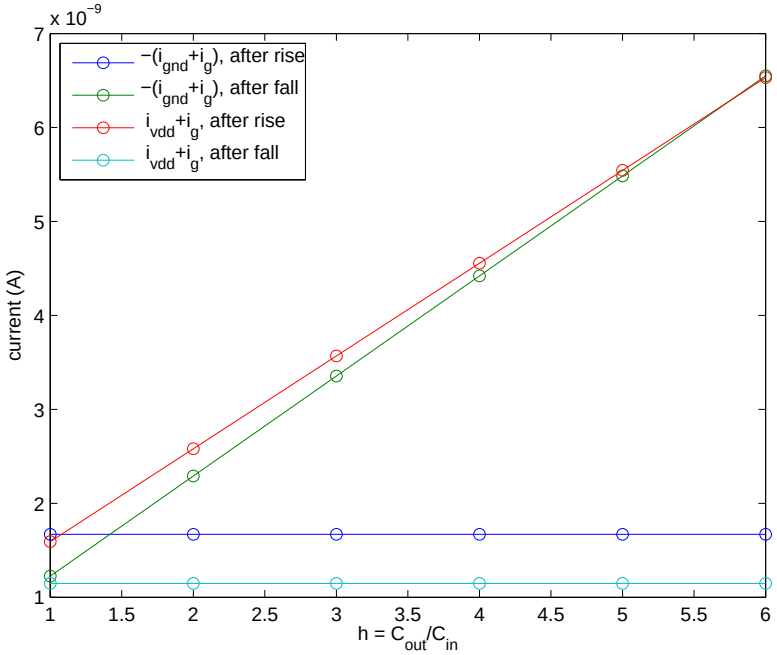

Figure 27. Leakage currents versus h for an inverter with $W_p/W_n = 3$ in a 65nm process. Shown are $-[i_{gnd} + i_g]$ and $[i_{vdd} + i_g]$ after a rise and a fall. Notice that the on network sources more current as h increases.

CHAPTER 4

LOGICAL EFFORT

4.1 RC Delay Model

For an arbitrary circuit element with a monotonic relationship between current and terminal voltage, $I = I(V_o, V_i)$, discharging a constant capacitance, for V_i constant with respect to time:

$$I(V_o, V_i) = C\frac{dV_o}{dt} \tag{28}$$

$$\int_{t_1}^{t_2} dt = \triangle t = C\int_{V_1}^{V_2} \frac{1}{I(V_o, V_i)} dV_o \tag{29}$$

$$R_{eff} = \int_{V_2}^{V_1} \frac{1}{I(V_o, V_i)} dV_o \tag{30}$$

$$\triangle t = CR_{eff} \tag{31}$$

Where the delay we are measuring is the amount of time it takes for the output voltage to go from V_1 to V_2. We find that delay scales linearly with C, and we can model the circuit element as an effective resistance, R_{eff}, so long as we do not change V_2 or $V_1$for which we are defining our delay.

This simple relationship breaks down when I becomes a function of input voltage and that input voltage a function of time, but this atleast motivates looking for regions in which delay is linear with capacitance.

Figure 28 shows the results of simulating an inverter and measuring average propagation delay (the ratio of the inverter was sized such that the rise and fall delays were relatively equivalent) versus input slew rate for various *electrical efforts* . For any given *electrical effort*, delay appears linear with input slew rate up until a point where it starts to

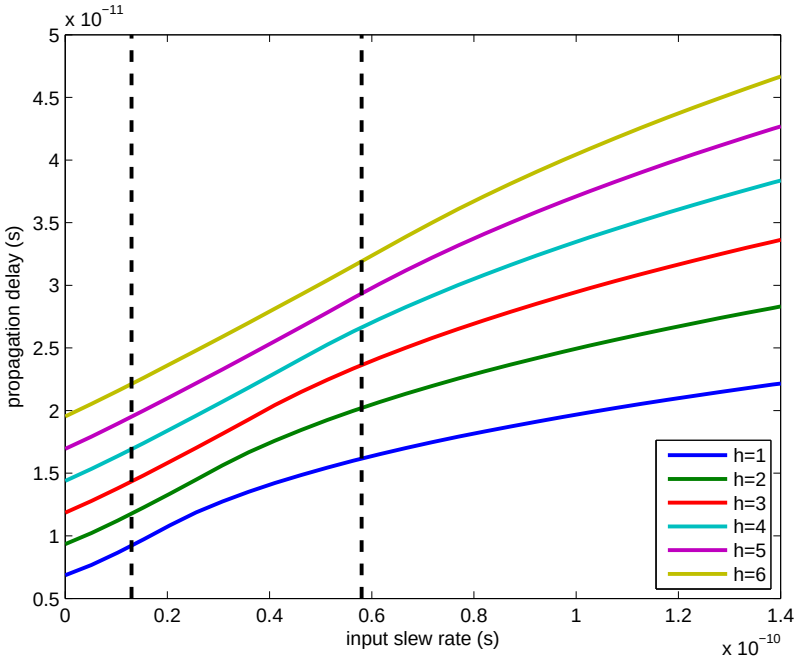

Figure 28. Average propagation delay of an inverter versus input slew rate for various *electrical efforts*. The leftmost vertical line shows the output slew rate of an inverter in a chain over inverters, all stages having $h = 1$. The right vertical line is the output slew rate of an inverter in a chain of inveters, all stages having $h = 6$. Simulation was carried out with BSIM 4v4 models from a commercial 65nm technology.

roll off, this point is increased with increases in *electrical effort*. For any given input slew rate, delay also appears linear with respect to *electrical effort* down to a point, where this point is decreased with decreases in input slew rate.

Figure 28 also has two vertical dashed lines showing two reference slew rates. The leftmost slew rate is that of the output slew rate of an arbitrary stage inverter in a chain of inverters, where each stage has $h = 1$. The right line represents the same measurement but for all stages having $h = 6$. Experiencing an input slew rate lower than the left line is not likely, and slew rates higher than the right line are again not likely for optimally sized designs. Within this range, delay appears relatively linear with respect to input slew rate and *electrical efforts* higher than one.

4.2 Logical Effort

The method of Logical Effort (LE) is used to predict the delay of CMOS logic gates and choose gate sizings to minimize delay along a path [7].

It defines the propagation delay of a gate as:

$$d_{abs} = \tau(gh + p) \tag{32}$$

Where the extrinsic parameter h is defined as $h = C_o/C_i$ and g, p are intrinsic gate dependent parameters. τ is a process and reference gate dependent variable with unit of seconds, and is derived from the reference gate, an inverter of a specific transistor sizing ratio for which $g \equiv 1$.

The product gh represents the delay associated with the gate driving subsequent gates' input capacitances, and p the delay that comes from the gate driving its own parasitic capacitances. Increasing the size of the driving gate increases its drive current, decreasing its propagation delay, but increases the load on the previous gate C_i. Assuming the gate's own parasitic capacitance increases at the same rate its driving current decreases, the delay from driving itself should remain relatively constant.

The delay p of a gate is then the delay a designer is stuck with when using that gate regardless of gate sizing, and is referred to as the *parasitic delay*. Intuitively, h is referred to as the *electrical effort* as increasing the load capacitance increases the amount of charge the gate has to move, decreasing the input capacitance reduces the speed by which the gate can move charge, and both represent an increase in h. Lastly, g is called the *logical effort* and is a parameter strongly dependent on the transistor topology of a gate required by whatever logic function the gate computes. A more complicated logic function, the more transistors, and generally the higher the delay.

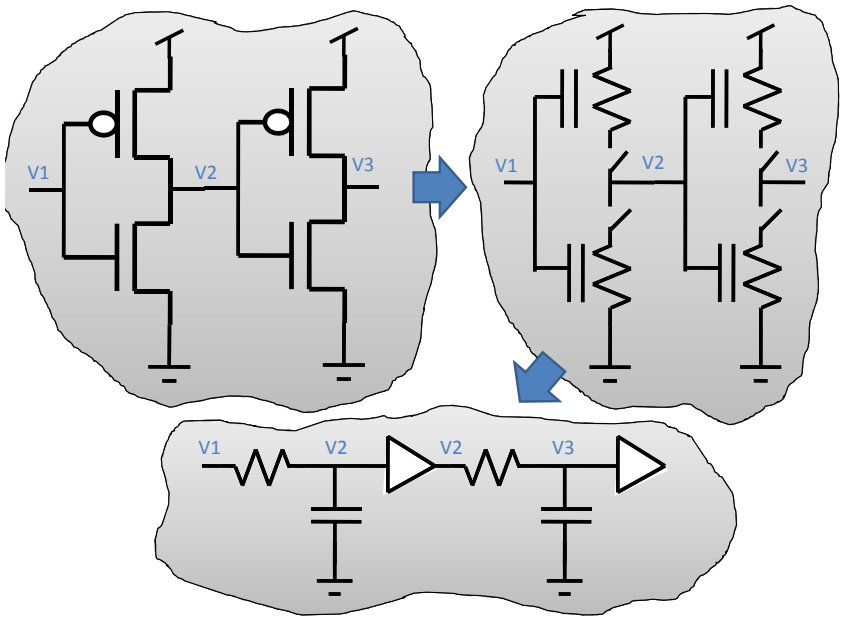

Figure 29. RC model for gate delay.

4.3 Simple Derivation

The method is based on a simple RC model for the delay of a gate, Figure 29, and differentiates between the delay associated with the gate driving subsequent load gate capacitances and the delay inherent in the driving gates' parasitic capacitance.

Start with:

$$d_{abs} \propto R_p C_l = R_p \left[C_p + C_o \right] \tag{33}$$

Where d_{abs} is the propagation delay across the gate, R_p an effective resistance of the driving network (pull up or pull down) over the entire input to output transition, C_p and C_o being effective parasitic and output load capacitances respectively.

Assuming R_p, C_p, and the input capacitance to the driving gate, C_i, scale with gate width, W, linearly, in the obvious fashion, write:

$$R_p = \frac{R_p'}{W}$$

$$C_p = C_p' W$$

$$C_i = C_i' W$$

Where the prime RC variables represent intrinsic parameters of a particular gate. With these write the original delay Equation 33 as :

$$d_{abs} = R_p' C_i' \frac{C_o}{C_i} + R_p' C_p' = \tau (gh + p) \tag{34}$$

Which, after some lumping of variables, minor algebra, and using the definition of *electrical effort*, produces 32.

CHAPTER 5

LOGICAL POWER

Power dissipated in a logic gate is a complicated function of the gate's implementation and its environment. The logic family, transistor topology, transistor sizings, and rail voltages that all define a gate are only part of the picture. The load that the gate drives, and transient inputs all factor into the power a gate will dissipate over time.

The power dissipated by the gate can be broken down into three well defined components: *dynamic power, static power,* and *short-circuit power.* Dynamic power is simply defined as the amount of power consumed by charging and discharging capacitances seen by the logic gate and can be expressed as such:

$$P_{dyn} = \alpha f V_{dd}^2 C_{tot} \tag{35}$$

Where $\alpha$is the activity-factor defining the fraction of the cycles that the logic gate makes an output transition, f the cycle frequency, $V_{dd}$the rail-to-rail voltage, and $C_{tot}$the total amount of capacitance seen by the driving gate.

Whereas *dynamic power* is dissipated during an output transition, *static power* is the power consumed when the gate is not making a transition and is a simple function of the static current draw I_{stat}:

$$P_{stat} = V_{dd} I_{stat} \tag{36}$$

The last portion of power, *short-circuit* power*,* is defined as the portion of power consumed during an output transition that did not go to charging capacitances. This portion of power is quite difficult to express compactly, and for even the most trivial of logic gates, an inverter, has no closed-form analytical solution. Most approximations are extremely cumbersome [3, 4, 2]. *Short-circuit power*, is a function of input slew rate, output slew rate, and gate topology:

$$P_{s.c.} = \alpha f V_{dd} F(input - transition, output - transition) \tag{37}$$

5.1 Components of Power

The idea of *Logical Power* is to define a small set of parameters for a given gate that predict these components of power as linear functions of $C_{in}$and C_{out}.

$$E_{tot} = x[\alpha((E_{dyn,vh} + E_{sc,vh})h + E_{dyn,h0} + E_{sc,h0}) + P_{stat}T] \tag{38}$$

$$E_{tot} = x[\alpha(E_{act,vh}h + E_{act,h0}) + P_{stat}T] \tag{39}$$

For each gate we need simply three new parameters $E_{act,vh}$,$E_{act,h0}$,P_{stat} to predict the energy consumption of that gate as a function of its size, output load size, cycle time, and activity factor.

5.1.1 Dynamic Energy

We start with *dynamic energy,* the portion of *dynamic power* that is the energy consumed given a transition, and write the total capacitance term as the summation of load capacitance C_L and parasitic capacitance C_P:

$$E_{dyn} = V_{dd}^2 C_{tot} = V_{dd}^2[C_p + C_{out}] \tag{40}$$

Plugging in h for $C_{out}$and grouping terms:

$$E_{dyn} = V_{dd}^2[C_p + hC_{in}] = x(E_{dyn,vh}h + E_{dyn,h0}) \tag{41}$$

Where $E_{dyn,vh}$, and $E_{dyn,h0}$are the intrinsic gate dependent terms that do not change with gate sizing, and x is a scaling factor such that $x = 1$ when the gate is minimum sized and $x = 2$ when the gate transistors are sized to be twice as wide as minimum for that gate.

$$E_{dyn} = xV_{dd}^2[C_p^{'} + h(1 + r)C_o] \tag{42}$$

5.1.2 Static Energy

Gate current aside, we would not expect *static energy* to depend on h and thus write:

$$E_{stat} = xV_{dd}i_{stat}T = xP_{stat}T \tag{43}$$

Where T is the amount of time that the gate spends in static operation.

5.1.3 Short-Circuit Energy

From [6] it can be seen that heavy approximations to *short-circuit* power are function of the input and output function slew rates. The idea being *logical power* is to approximate *short-circuit power* as a simple function of propagation delay.

If we assume a simple delay based, linear predictor for *short-circuit energy*, we can approximate it as such:

$$E_{sc} = F(v_i(t), v_o(t)) \doteq F(d) = F(gh + p) = x(E_{sc,h}h + E_{sc,h0}) \tag{44}$$

It is often convenient to lump *short-circuit energy* and *dynamic energy* together, as they both occur on output transitions, and especially since *logical power* is predicting them with simple linear functions of h :

$$E_{act} = E_{dyn} + E_{sc} = x[(E_{dyn,vh} + E_{sc,vh})h + E_{dyn,h0} + E_{sc,h0}] \tag{45}$$

$$E_{act} = x[E_{act,vh}h + E_{act,h0}] \tag{46}$$

Where $E_{act}$is the energy consumed by the gate on an output transition independent of activity factor and cycle time.

CHAPTER 6

METHOD VERIFICATION

6.1 Capacitance Estimation

The input capacitance, C_{in}, of a logic gate is the summation of all of the capacitances looking into the gates of the transistors connected to a particular input. While the gate-source/drain overlap capacitances are relatively constant, the gate-channel capacitance changes significantly depending on the mode the transistor is in. To complicate things further, the Miller Effect can become quite significant depending on the speed by which the sources and drains change voltage.

These contribute to causing $C_{in}$to be a function of time over input and output transitions. Finding a closed form expression for$C_{in}(t)$ is both difficult and unnecessary for these gate characterization methods. Instead we look for an *effective* capacitance, $C_{effective}$, defined as such:

$$C_{effective} = \frac{1}{V}\int idt \tag{47}$$

Where the limits of the integral are over some range of extraction, and V is the difference between the voltage at those same limits. For instance, in calculating C_{in} integrate over when the input current starts to change to when it stops changing by some arbitrarily small percentage, and V to be V_{dd}.

Then define C_o to be equal to the effective capacitance looking into the input of a minimum sized inverter, of transistor widths $W_{min}$and length L_{min}, divided by two over an input transition:

$$C_o = \frac{1}{2V_{dd}}\int idt \tag{48}$$

Short-channel and fringing effects aside, gate capacitance should scale linearly with the product of W and L, such that:

$$C \propto WL \tag{49}$$

$$C_{in} \simeq \frac{2WL}{W_{min}L_{min}}C_o \tag{50}$$

In calculating *electrical effort*, h, we do not even need $C_o$as it always cancels out:

$$h = \frac{C_{out}}{C_{in}} = \frac{\sum W_o}{\sum W_i} \tag{51}$$

Where the numerator is the summation over the widths, W_o , of all of the transistors whose gates are connected to the output of the driving gate, and the denominator summation is over all transistors, W_i , connected to a particular input of the driving gate. This assumes that all transistors have the same length.

Define $r = W_p/W_n$to be ratio of of the widths of the pMOS and nMOS transistors whose gates are connected to a particular input, and $x = W/W_{min}$to be the gate's scaling factor where all gate transistors are scaled by x relative to some minimum W_{min}.

$$C_{in} = x(r + 1)C_o \tag{52}$$

To evaluate the magnitude of the error in the assumptions made in 50and 52simulations of a five deep inverter chain of $h = 1$ were made. $C_{in}$was extracted from the fourth inverter by applying 47to its load current. The x and r of all gates in the chain were varied and the results can be seen in 30.

Figure 31 shows the when extracting $C_o$and using Equation 52to estimate C_{in}. The error over these ranges ($r = 1 : 3$, $x = 1 : 4$) is less than 3.5%. The error consistently being that the the estimation underpredicted C_{in}.

6.2 Statistical Path Analysis

To evaluate the accuracy of *logical power* randomized test circuits were created. The test circuit was a 12 stage logic path comprised of two to four input NANDs and NORs as well

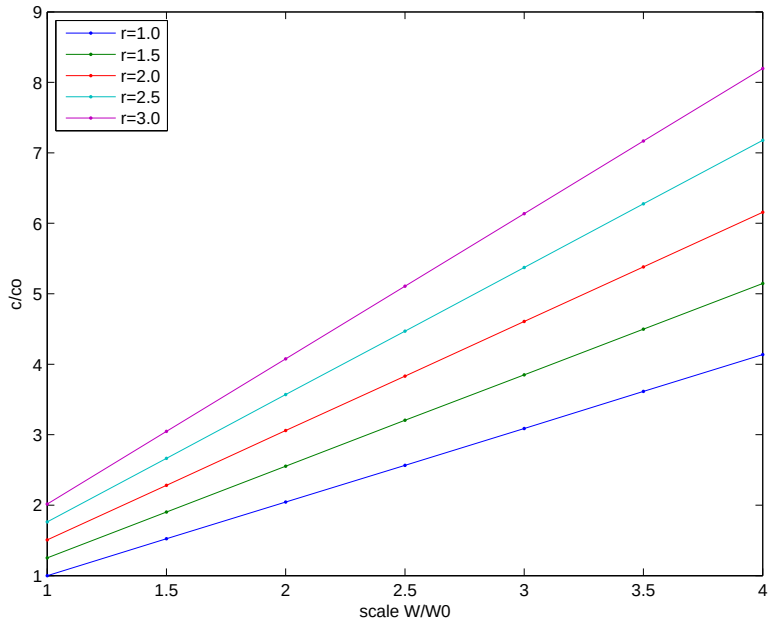

Figure 30. Extracted effective normalized input capacitance versus $x = W/W_0$ for various $r = W_p/W_n$ for an inverter simulated with a 130nm BSIM 3v3 model.

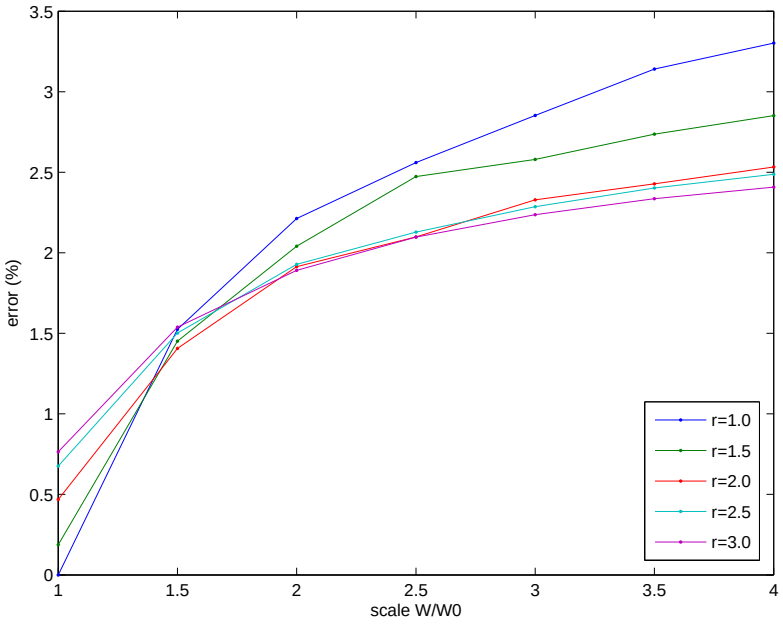

Figure 31. Estimated capacitance error versus extracted capacitance versus$x = W/W_0$ for various $r = W_p/W_n$ for an inverter simulated with a 130nm BSIM 3v3 model.

Table 2. Statistical results for Logical Power

Tech	Mean Error (%)	Max Error (%)	Active (%)	Static (%)
130nm	0.7	4.5	99	1
90nm	5.3	10.2	84	16
65nm LP	10.4	18.4	99	1
65nm SF	6.0	9.5	77	23

as inverters all scaled to random sizes between 1 and 8x minimum size.

This type of test circuit was chosen because each gate has an activity factor $\alpha_i = 1$. In general on the most trivial of circuits have this property, and the individual activity factors being dependent on circuit topology and input vector. This circuit was chosen to evaluate the effectiveness of *logical power* in the limit of perfect knowledge of activity factor, as the method itself is independent of whatever system one uses to predict activity factors.

Propagation delay and total power was measured from the 3rd to 10th stage, as well as predicted by *logical power* and *logical effort* and the results were compared. Out of 1000 randomly generated circuits, *logical effort* had an average error of 0.8% while *logical power* an error of 1.6% with a worst case of 5%. Considering that capacitance estimation is off by 3% for $x = 4$ and only gets worse the higher x is, capacitance estimation error is probably the dominant source of error in Logical Power.

As a comparison, if for each circuit the average energy consumption of the 1000 random circuits was used as a predictor, the average error was 10% with a worst case of 35%.

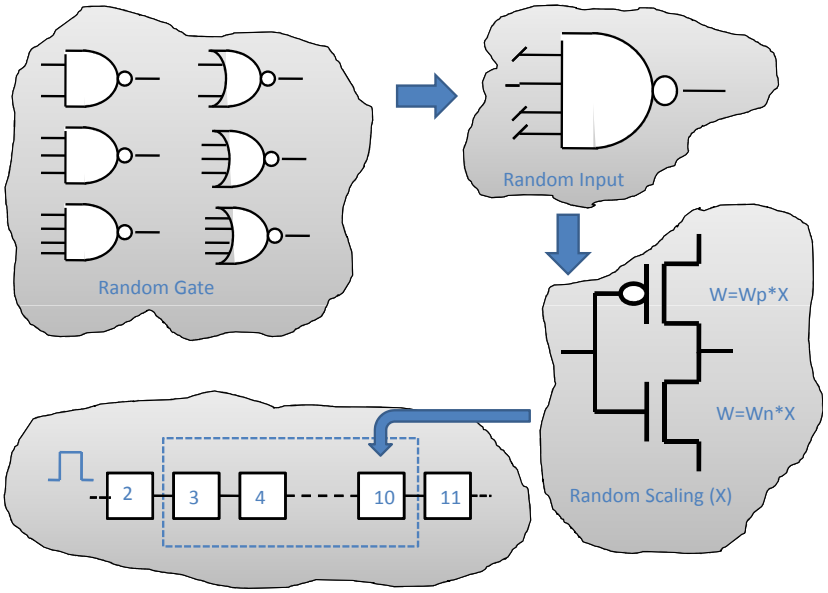

Figure 32. Randomly generated test circuit. A pool of 2 to 4 input NANDs and NORs and inverters of random sizes (1 to 8x) are chosen to comprise the 12 stage circuit. Propagation delay and energy consumption of the 3rd to 10th stage are measured in simulation and predicted with Logical Effort and Logical Power.

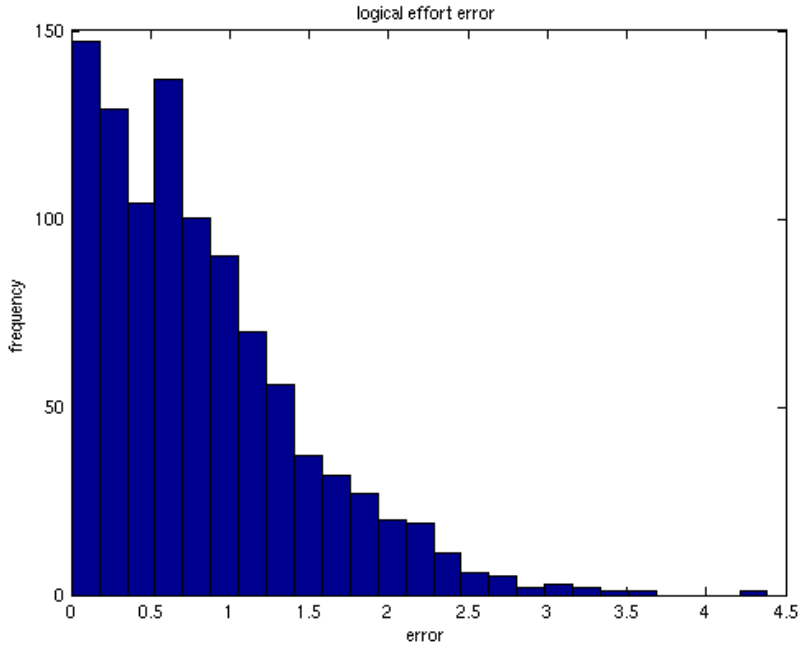

Figure 33. 1000 Random test circuits, Logical Effort Error. Mean 0.8%.

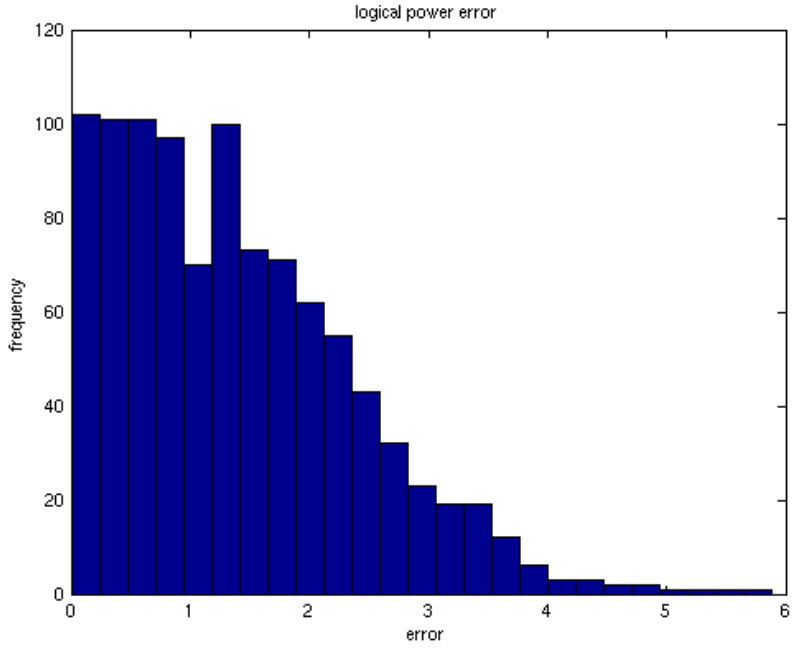

Figure 34. 1000 random test circuits. Logical Power error, mean 1.6%.

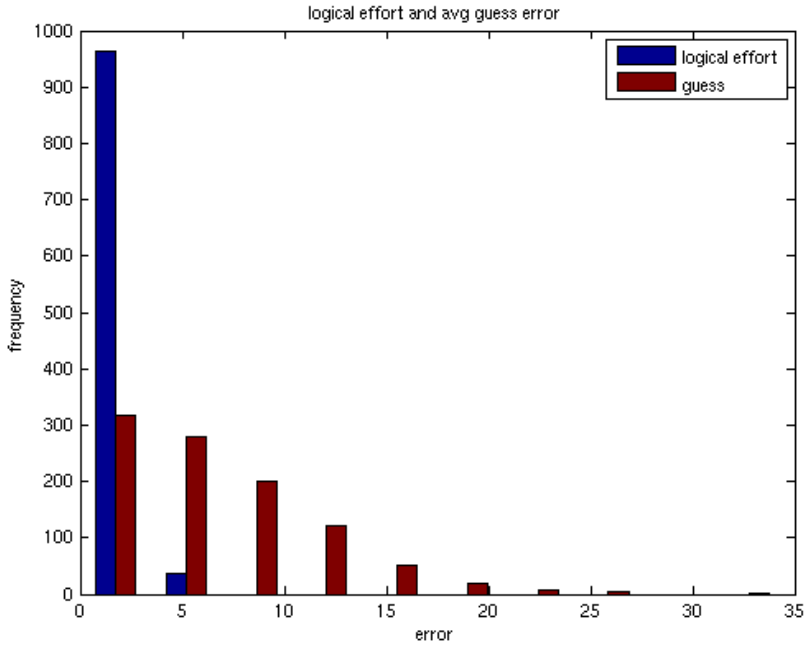

Figure 35. Logical Power is compared to predicting energy as the average energy consumption of all 1000 test circuits.

CHAPTER 7

PARAMETER EXTRACTION

To extract the *Logical Power* parameters we need to be able to accurately measure the components of power for a given logic gate versus *electrical effort*. To do so we employ the same characterization structure used to extract the parameters for *Logical Effort*.

Figure 36 shows the circuit used for gate characterization. It is simply the circuit suggested in chapter 5 of [7]. The circuit comprises multiple chains of gates, where each gate is the same gate to be characterized. The chains are five stages deep. In any chain each stage is subjected to the same *electrical effort* with varying *electrical efforts* between chains.

A pulse is applied to the first stage and performance is measured on the third stage, the first two stages used to shape the input pulse to something reasonable. Integer values of *electrical effort* are achieved by loading a gate with multiples of itself, each load being loaded to reduce Miller capacitance.

7.1 Logical Effort

Logical Effort predicts delay as:

$$d_{abs} = \tau(gh + p) \tag{53}$$

Shown in figure 37the input and output voltages to the third stage of six different paths of stage electrical efforts $h = 1$ through $h = 6$ for the same two-input NAND gate. This particular simulation is of the propagation delay of the a-input and shown is the rising output transition.

The next plot, figure 38, is of d_{abs} extracted in the above mentioned manner versus h. Rising, falling, and average logical effort and parasitic delay can be extracted from this plot by:

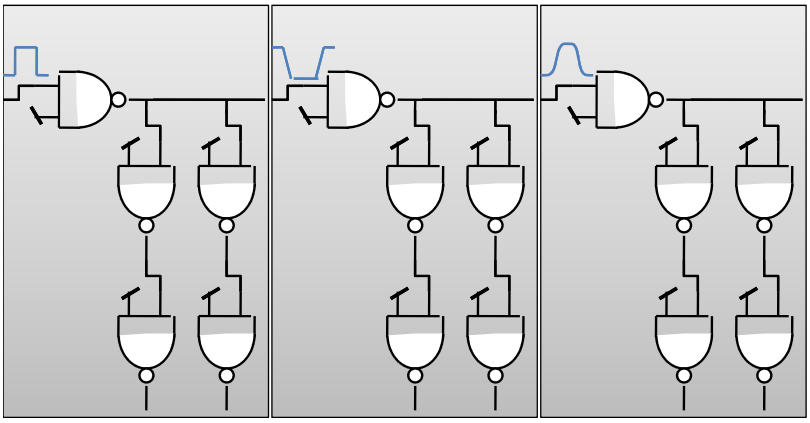

Figure 36. Characterization circuit.

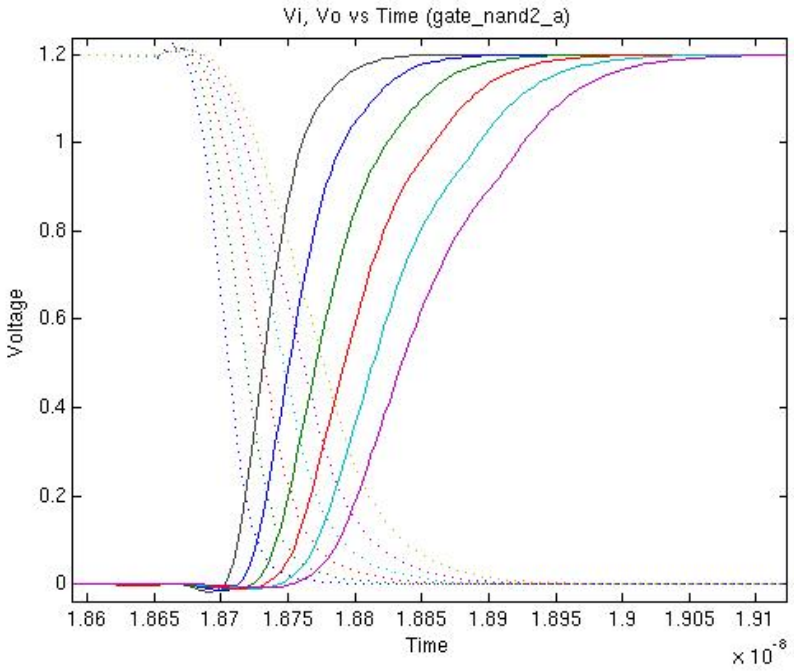

Figure 37. Input and output voltages versus time for input-a of a 2-input NAND gate for various *h*

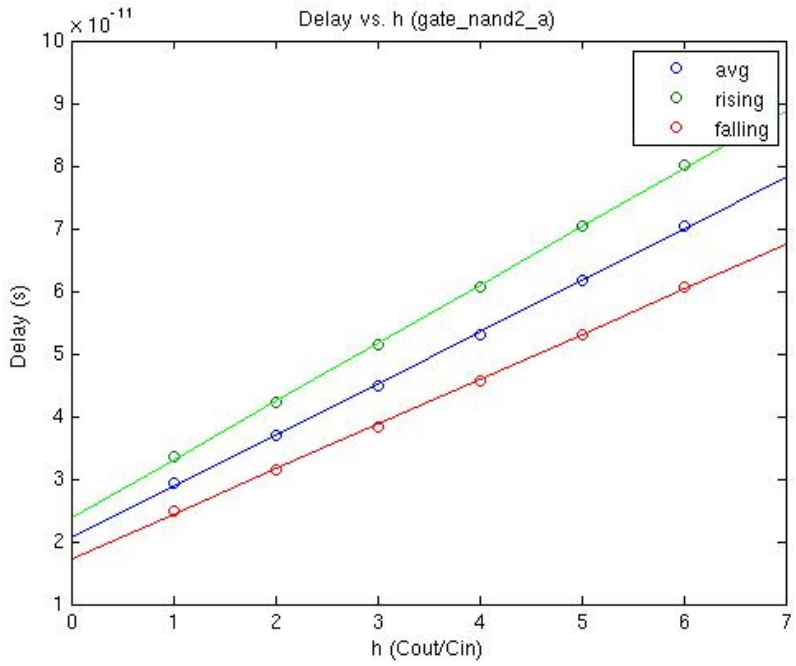

Figure 38. Rising, falling, and average propagation delays versus h for input-a of a 2-input NAND

$$\frac{d}{dh}d_{abs} = \tau g \tag{54}$$

$$d_{abs}(h = 0) = \tau p$$

Where τ is extracted from a reference inverter for which $g_{avg} \equiv 1$.

As is generally the case for non symmetric input gates, the parameters of *logical effort* will depend on which input is being characterized. The data from figure 38 would have been used to characterize the a-input and was done so by setting the b-input to VDD for the entirety of the simulation, from this we would obtain values for $g_{r,a}$, $g_{f,a}$, $p_{r,a}$, and $p_{f,a}$. Figure 39 shows this for both a and b inputs of the 2-input NAND gate.

7.2 Logical Power

As is the case with *Logical Effort*, the parameters of *Logical Power* generally depend on which gate input is making the deciding transition. So all parameters are extracted per input

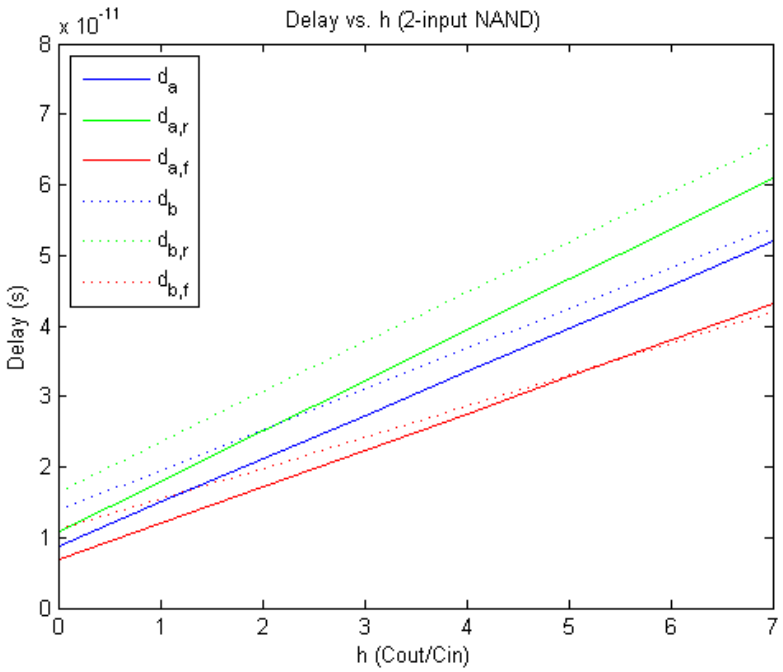

Figure 39. Rising, falling, and average propagation delays versus h for both the a (solid lines) and b (dotted lines) inputs of a 2-input NAND

using the same circuit as before.

Figure 40 shows the integrals of i_{vdd} and i_{gnd} versus time for various h. Using equations 10 13 19 24to extract the components of power versus h as shown in figure 41. The parameters of *logical power* are extracted by taking the derivatives with respect to h.

As shown in figures 4 and 40 there are points that define when the circuit moves from an active mode of operation to a static mode. Since this is a process governed the charging of capacitances, and therefore the voltages only become completely static at $t \rightarrow \infty$ we must make an arbitrary choice for how we define this point. Many methods were tried, and the method which seemed to be the most reliable without being too computationally significant, was to simply find the point in time where the leakage currents were within a factor of two of what they eventually settle to in the simulation time frame.

From Figure 42 it can be seen that *short circuit energy* is not very linear with h , and therefore Equation 44 is a rather poor predictor. However, from Figure 41 it can be seen

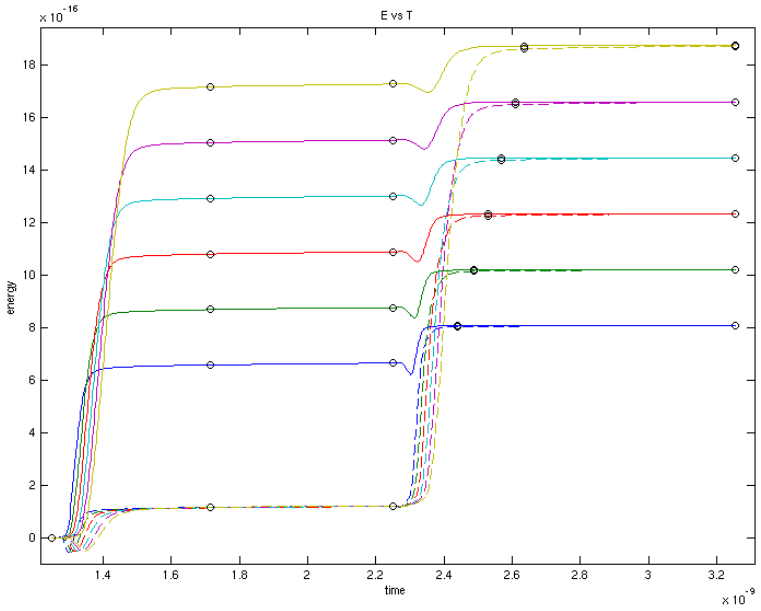

Figure 40. Integrals of i_{vdd} (solid lines) and i_{gnd} (dashed lines) versus time for $h = 1, 2, 3, 4, 5, 6$ for a 3-input NAND gate, characterizing the second input. A BSIM 4v4 model file from a commercial 65nm low power logic process was used.

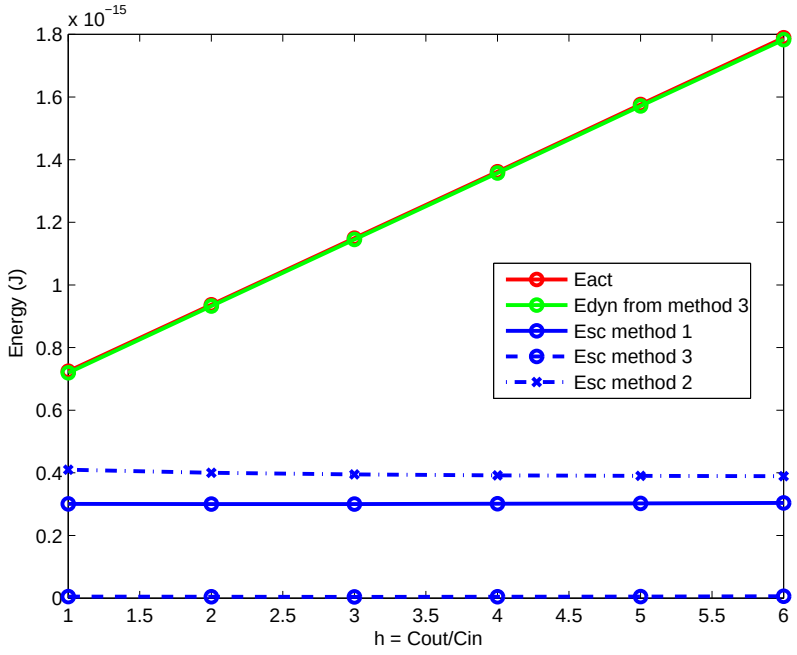

Figure 41. Extracted h dependent energy for the b-input of a 2-input nand gate. Shown are active, dynamic as extracted from active by the short circuit method 3, and short-circuit energy by each method.

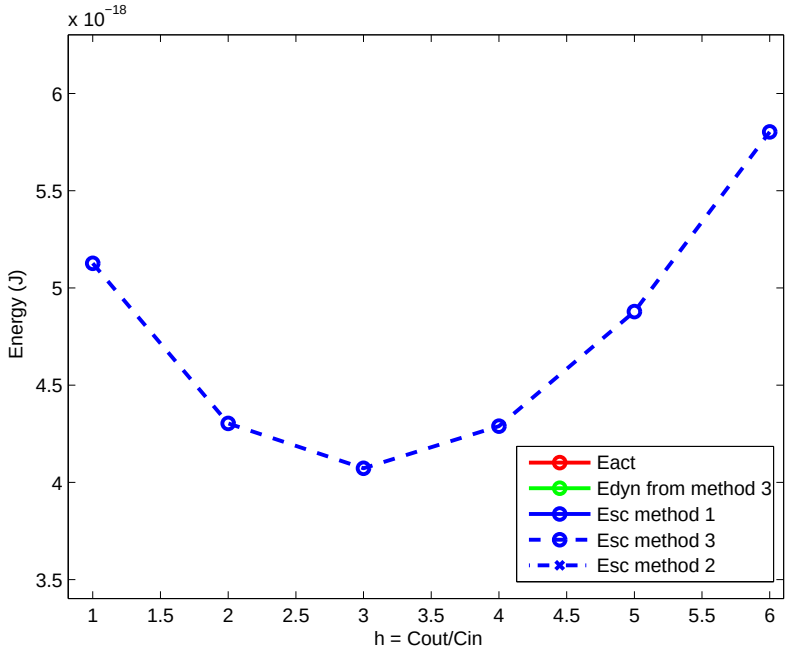

Figure 42. Extracted h dependent energy for the b-input of a 2-input nand gate. This is a zoomed in plot of the data in Figure 41 in order to show *short circuit energy*.

that *short circuit energy* is a very small percentage of *active energy* for a 2-input nand gate. It turns out that it is a very small percentage of total energy for a wide range of cases and therefore *active energy* appears quite linear with *h*.

CHAPTER 8

CONCLUSION

A new method for measuring *short circuit* energy is proposed that has a much more intuitive response to input slew rate and output load size than previous methods. Previous methods could also report nonsense negative numbers for *short circuit* energy, or would not trend to zero for step inputs. The new method elegantly avoids both of these problems. The new method also suggests that *short circuit* energy is a much smaller percentage of total energy than previously thought. It shows, as previous methods did, that *short circuit* energy is highly dependent on input slew rate.

Logical Power was proposed as a simulation based extraction methodology to characterize and predict the power of a logic gate. It introduces three new gate dependent parameters, $E_{act,vh}$, $E_{act,h0}$, and P_{stat}, that along with the parameters of *Logical Effort*, g, and p allow for the accurate prediction of a gate's energy consumption and propagation delay:

$$E_{tot} = x[\alpha(E_{act,vh}h + E_{act,h0}) + P_{stat}T] \tag{55}$$

$$d_{abs} = \tau(gh + p) \tag{56}$$

Where the variables x, the unit less scaled width of the gate, and h the ratio of load capacitance to input capacitance, represent the gate's physical environment.

Logical Power, like Logical Effort, ignores input slew rate in prediction, but this is shown to produce trivial amounts of error over a wide variety of test cases where both methods are generally higher than 90% accurate, and usually closer to 99% accurate. It is suggested that the main source of error be in the capacitance estimation techniques, that is, assuming a $x = 2$ gate to have twice the capacitance of the $x = 1$ case when it is not.

CHAPTER 9

CHARACTERIZATION SOFTWARE

The software is readily available through a Subversion repository .

We want to characterize a logic gate. Take for example, this two-input NAND gate.

9.1 Characterization Circuit

In order to extract the parameters of Logical Effort and Logical Power the gate to be characterized will be placed in a characterization circuit for simulation. The circuit is composed of many paths constructed out of the gate. Each path has five stages, and the electrical effort of each stage is constant within a path, with each path having a different electrical effort for each stage.

Shown is the first stage of the first two paths. In the first path $h = 1$, in the second $h = 2$. This structure is particularly convenient in that we do not actually need to know the absolute values of the output and load capacitances to get h as we've set it up to be the ratio of multiples of the same capacitance.

Measurements will actually be performed on the third stage of each path, with the first two stages shaping the input pulse into something reasonable. Gates designed to be loads, themselves are loaded, otherwise their outputs would swing unrealistically fast skewing their own input capacitance. The measurements will be input and output voltages, gate input, output, vdd, gnd, bulk, and well currents as a function of time.

9.2 Installation and Setup

From a command prompt:

svn checkout svn+ssh://<you>@ecelinsrv2.ece.gatech.edu/tools/cadsp/svn/mad/logical_power/trun
logical_power

cd matlab

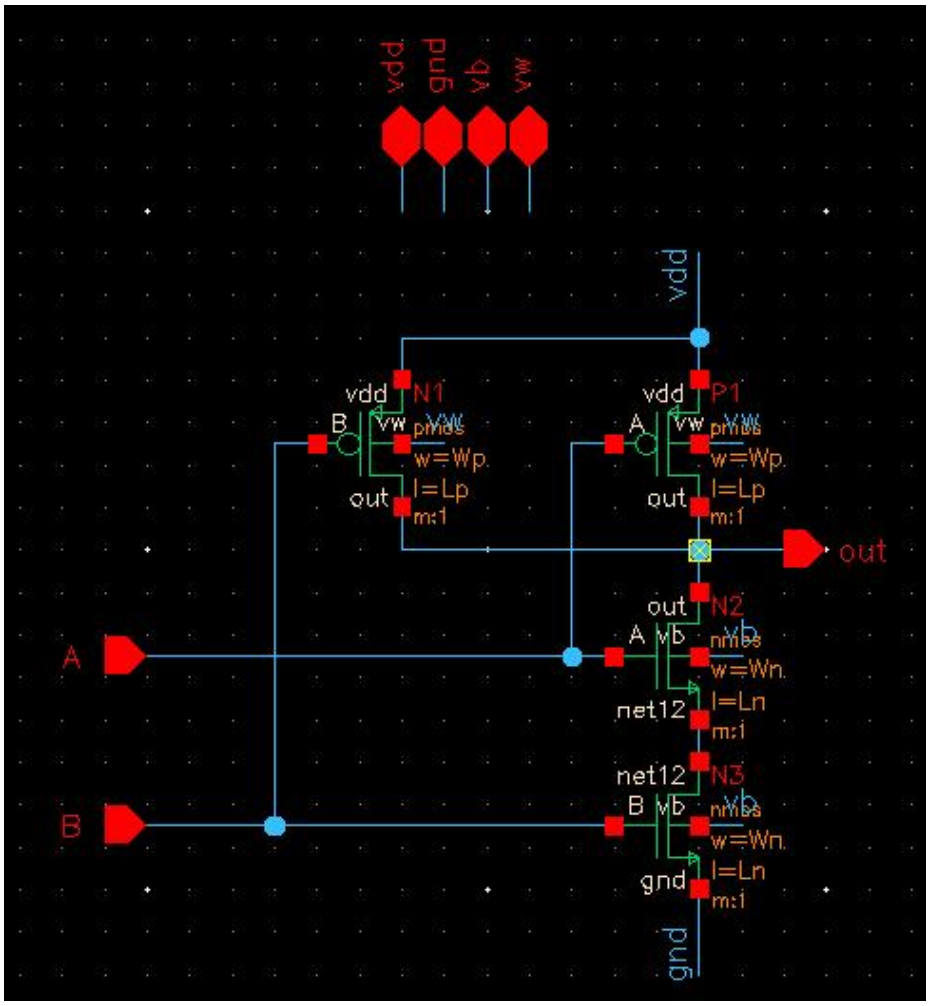

Figure 43. 2-input NAND gate

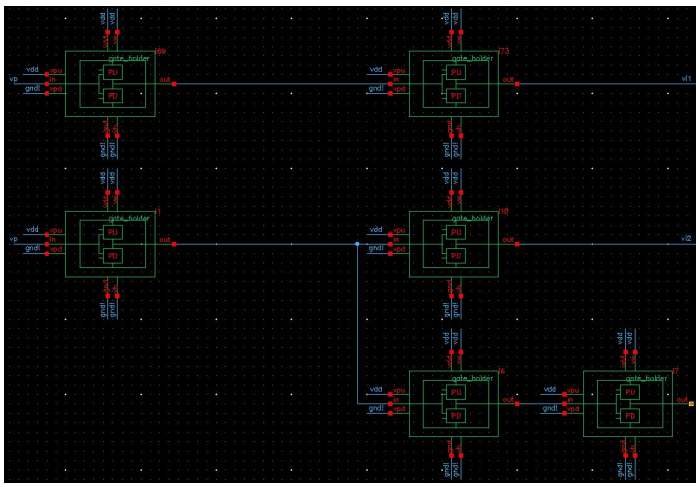

Figure 44. gate char stage one

matlab

At this point, the Cadence side of the software suite should be working as well. You can try:

cd cadence

source cshrc.ncsu61.ece.lin

virtuoso

9.3 Simple Example

Start Matlab and then try:

>> inv = gateChar('ibm_130n.scs');

This sets up a simulation in a 130n technology defined by the file ibm_130n.scs. A lot of defaults have been set at this point to quickly get to simulating. Check around in the inv.params.cir, inv.params.dim, inv.params.tim, inv.params.ana structures. For instance:

>> inv.params.cir

ans =

tech_file: '"ptm_130n.scs"'

gate_to_char: 'gate_inv'

circuit_file: 'gateChar_6h.scs'

gates_file: '"gates.scs"'

raw_out_file: 'temp.out'

simulator: 'spectre'

VDD: 1.2000

>> inv.params.dim.W

ans =

n: 1.8000e-07

p: 5.4000e-07

The default gate to simulate is an inverter, labeled gate_inv, where the gate_ terminology means we've wrapped the gate into a universal gate holder that can be plug and placed with any other gate into the characterization circuitry.

The default voltages and dimensions are set based on the technology file used. From here we can immediately simulate:

>> inv.runSim();

When this completes we can quickly generate some plots by:

>> inv.plotD

>> inv.plotI

>> inv.plotQ

Or we can access the data structures in the class gateChar directly to generate less obvious plots. For more information on the data structures, see logical_power/matlab/@gateChar/gateChar.m and take a good look at the comments in the properties section especially regarding the properties rawData, extData, and anaData.

9.4 Modifying Default Parameters

There are a pile of default parameters set by the gateChar constructor.

Anywhere in any of the circuit files (gates.scs, tech_file.scs, gate_char.scs) variables can be made and set by matlab. The syntax for the variable placeholder in the circuit as well as the structure definition for setting the variable in matlab is:

in gates.scs

...

x0 1 0 res <resx>

in matlab

>> x = gateChar('tech_lib.scs');

>> x.params.some_category.res.x = val;

>> x.runSim();

Where some_category is any name you want it to be, its simply for organizational purposes. dim holds transistor dimensions, cir holds gate type, vdd, etc, tim holds timing information and so on. Evertying to the right of some_category gets turned into the variable name to search for. In this case, res.x gets turned into resx. Again this is just for convenience and the simulator supports any number of nestings in the structure.

When runSim is called it builds a new circuit file with the variables specified. The resulting circuit file will have <resx> replaced with val when simulated.

9.5 Sweeps

The sweeps functionality allows one to sweep parameters and characterize tweaked versions of a base gate. The syntax is as follows:

x = gateChar('tech.scs');

x.params = some_struct;

x.sweeps(1).some_category.vector_11;

x.sweeps(1).some_category.vector_12;

...

x.sweeps(2).some_category.vector_1n;

x.sweeps(2).some_category.vector_21;

x.sweeps(2).some_category.vector_22;

...

x.sweeps(2).some_category.vector_2m;

...

x.sweeps(N).some_category.vector_N1;

x.sweeps(N).some_category.vector_N2;

...

x.sweeps(N).some_category.vector_Nz;

x.runSim();

Where some_struct contains whatever initial parameters you want to set. And the vectors are values to be swept. All vectors within sweeps(1) will be swept together and versus all vectors in sweeps(2) versus all vectors in sweeps(3) versus ... all vectors in sweeps(N).

The software exploits the parallelization of the parametric simulation on multi-core systems. Each sweep point is a new SPICE simulation independent of the previous, all SPICE simulations are setup in advance and then dispatched to as many cores as the system has access to (the current version of MATLAB 2009a supports a maximum of eight cores).

9.6 2D Sweep Example: r, VDD

Lets try the following code:

>> inv2 = gateChar('ptm_130n.scs'); %default is inverter

>> inv2.sweeps(1).dim.W.p = [1:0.5:2]*inv2.params.dim.W.n;

>> inv2.sweeps(2).cir.VDD = [0.8, 1.2];

>> inv2.runSim();

From there we've told the suite that we want to characterize this gate for an array of different parameters. The gate will be characterized for the following 2d space, various r = Wp / Wn ratios and various VDD values. The results can then be accessed and viewed in various ways.

>> inv2.plotD({1,1}) %plots delay for r = 1.0 vdd = 0.8

>> inv2.plotD({2,2}) %plots delay for r = 1.5 vdd = 1.2

In general, a netlist of gates for possible characterization is defined. These gates can have many variables associated with them, for instance, widths and lengths of transistors, operating voltages, etc. In fact, in the gate characterization netlist, the gate to be characterized is itself a variable as well as the technology file defining the transistor models.

In Matlab you set up a simulation by defining what all of these variables are, Matlab then parses the netlist replacing the variables with actual values and calls Spectre to simulate the netlist, Matlab then parses the output and analyzes the data produced, extracting all of the relevant gate characterization parameters.

Matlab will also set up and simulate an arbitrary dimension of sweeps, 3d, 4d, ... Nd there's no limit.

9.7 1D Sweep Example: r

However, lets explore 1d with one more example. Sweeps make it easy to do things like find the r that minimizes gavg for a variety of gates, etc. For instance:

>> r = [1:0.1:3];

>> inv3 = gateChar('ibm_130n.scs');

>> inv3.sweeps(1).dim.W.p = r*inv3.params.dim.W.n;

>> inv3.runSim(); >> figure, plot(r, inv3.extData(:,1:6), '-');

>> xlabel('Wp/Wn'); >> ylabel('g,p');

>> legend('g_{avg}', 'g_r', 'g_f', 'p_{avg}', 'p_r', 'p_f');

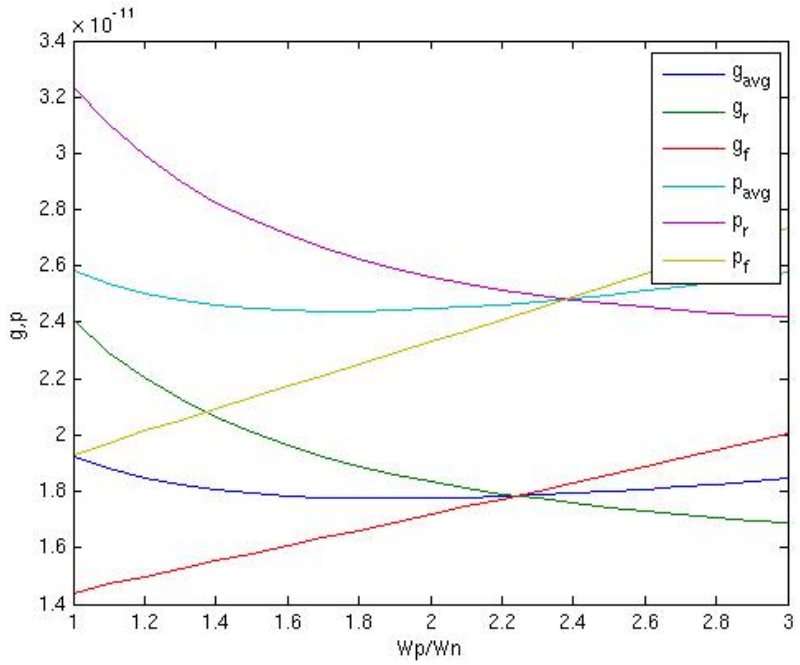

Figure 45. ***Logical effort*** **and** ***parasitic delay*** **for an Inverter versus** $r = W_p/W_n$

And here's the resulting plot to the right. Note that this is a good example of why equal rise and fall times are unobtainable for arbitrary load sizes, as when $g_r = g_f$ $p_r \neq p_f$.

Netlists describing the gates can be generated by hand, or graphically / schemactically through Cadence which will be the topic of the next section.

9.8 Generating Netlists

Ok, so you've got some gates in mind that you want to characterize, and you don't want to write netlists by hand. Here's what we do. We're going to generate a gates netlist file like the one located in

logical_power/matlab/circuits/gates.scs

So open it up, check it out, and keep it in mind as we go through the following. Move to the cadence directory, then:

source cshrc.ncsu61.ece.lin virtuoso &

Open up the library manager (tools -> library manager) and go to the logical_power library. We'll be concerned with the following cells:

- GATES
- GATE_CHAR
- gate_nand2_a
- nand2

Open up GATES, this is the schematic that will hold all of the gates that we hope to characterize. Notice that they all have the same ports. Each gate at this view represents a particular input of a particular gate to be characterized. For instance, there are two for the nand2 gate, one for input a and one for input b. If you'll notice, all gates have the same number of ports so that they can be exchanged without headache into the gate characterization circuit. Now descend all the way down to the nand2 schematic by way of the gate_nand2_a schematic.

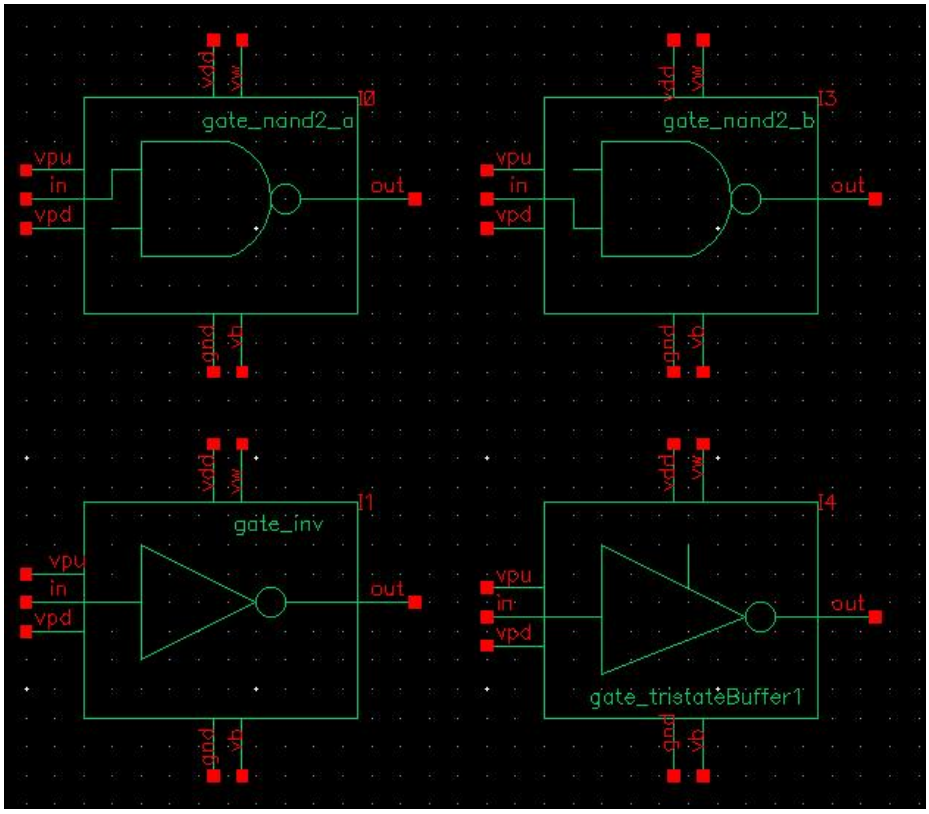

Figure 46. Gates to be characterized

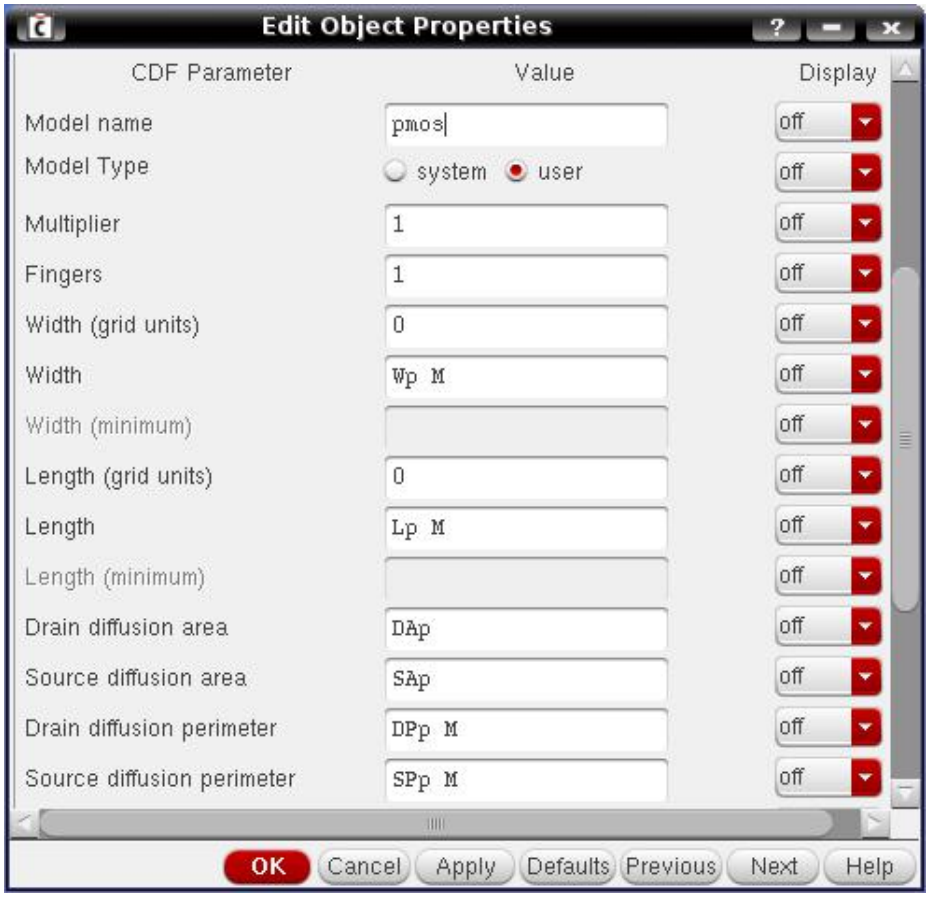

Figure 47. PFET properties

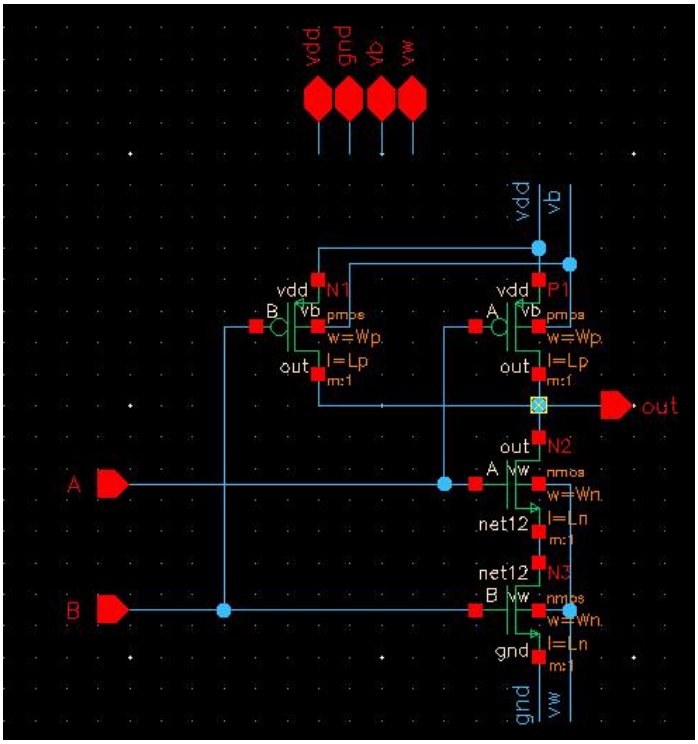

Figure 48. 2-input NAND gate

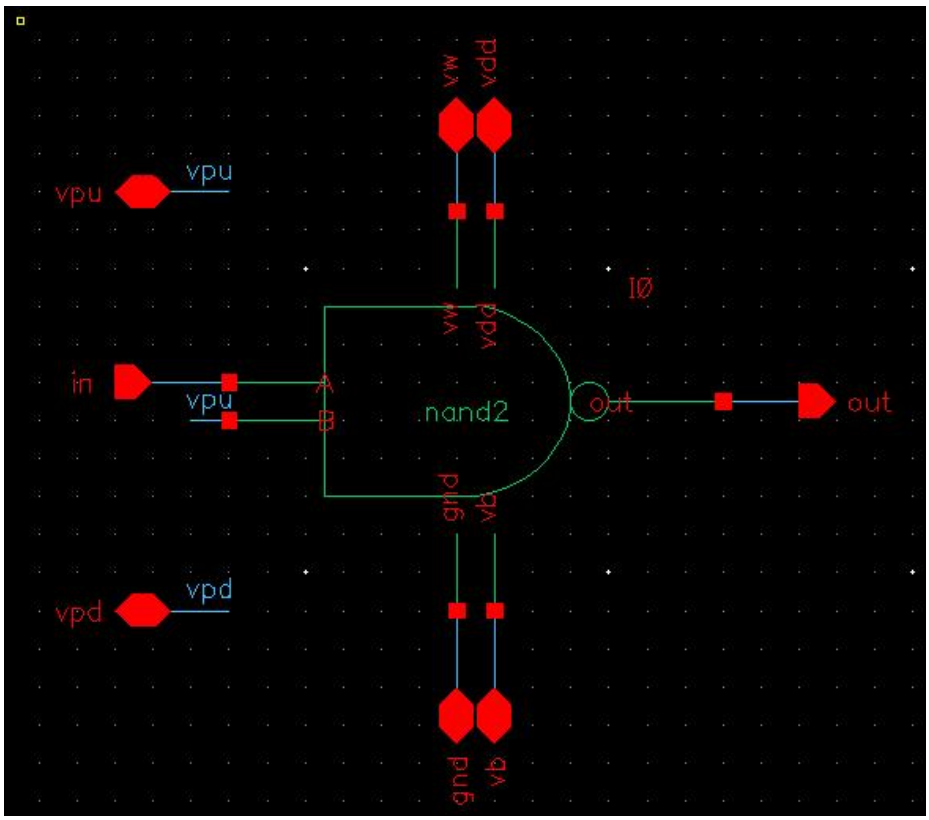

Figure 49. NAND2 input-a

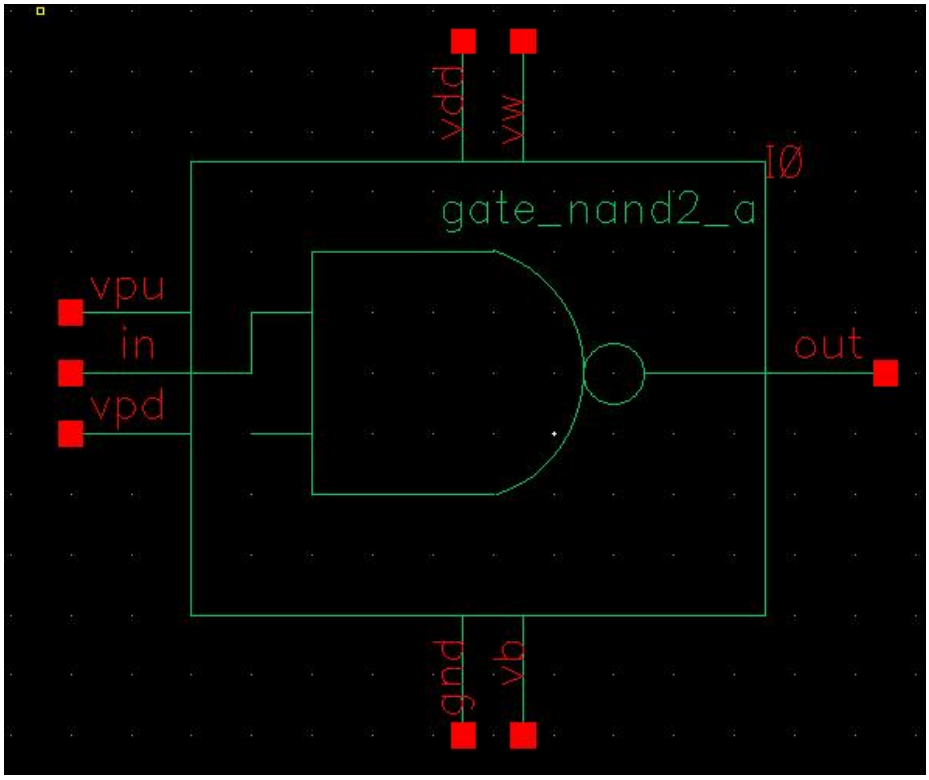

Figure 50. input-a of a NAND2 gate

REFERENCES

[1] S. M. Kang, "Accurate simulation of power dissipation in vlsi circuits," *IEEE JOURNAL OF SOLID-STATE CIRCUITS*.

[2] L. Bisdounis and O. Koufopavlou, "Short-circuit energy dissipation modeling for submicrometer cmos gates," *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS I: FUNDAMENTAL THEORY AND APPLICATIONS*, vol. 47, pp. 1350–1361, 2000.

[3] H. J. M. Veendrick, "Short-circuit dissipation of static cmos circuitry and its impact on the design of buffer circuits," vol. 19, no. 4, pp. 468–473, 1984.

[4] N. S. Srinivasa R. Vemuru, "Short-circuit power dissipation estimation of cmos logic gates," *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS–I: FUNDAMENTAL THEORY AND APPLICATIONS*, vol. 41, pp. 762–765, 1994.

[5] W. Grabinski, "Ekv model v2.6 and extraction methodologies," 2001.

[6] T. S. Koichi Nose, "Analysis and future trend of short-circuit power," *IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS*, vol. 19, pp. 1023–1030, 2000.

[7] D. H. Ivan Sutherland, Bob Sproull, *Logical Effort: designing fast CMOS circuits*. Morgan Kaufmann, 1999.